Universidad Nacional Mayor de San Marcos

Universidad del Perú. Decana de América

Dirección General de Estudios de Posgrado

Facultad de Ingeniería de Sistemas e Informática

Unidad de Posgrado

SOTESTER – Sistema de recomendación de técnicas de testing de software: Un enfoque colaborativo

TESIS

Para optar el Grado Académico de Magíster en Ingeniería de Sistemas e Informática con mención en Ingeniería de Software

AUTOR

Ronald Eduardo IBARRA ZAPATA

ASESOR

Glen Darío RODRÍGUEZ RAFAEL

Lima, Perú

2019

Referencia bibliográfica

Ibarra, R. (2019). *SOTESTER – Sistema de recomendación de técnicas de testing de software: Un enfoque colaborativo*. Tesis para optar el grado de Magíster en Ingeniería de Sistemas e Informática con mención en Ingeniería de Software. Unidad de Posgrado, Facultad de Ingeniería de Sistemas e Informática, Universidad Nacional Mayor de San Marcos, Lima, Perú.

---

1. Código ORCID del Autor: 0000-0002-5954-6265
2. Código ORCID del Asesor: 0000-0002-4471-3198
3. Grupo de Investigación: Ninguno
4. Institución que Financia la Investigación: Ninguna
5. Ubicación Geográfica: Lima – Perú, 12.0464° S, 77.0428° W
6. Año o rango de Años que abarcó la investigación: 2016-2017
7. DNI: 41302993

Universidad Nacional Mayor de San Marcos
Universidad del Perú. Decana de América
Facultad de Ingeniería de Sistemas e Informática
Vicedecanato de Investigación y Posgrado
Unidad de Posgrado

SUSTENTACIÓN DE TESIS PARA OPTAR EL GRADO ACADÉMICO DE MAGÍSTER EN INGENIERÍA DE SISTEMAS E INFORMÁTICA CON MENCIÓN EN INGENIERÍA DE SOFTWARE

En la Ciudad Universitaria, a los dieciséis (16) días del mes de enero del 2020, siendo las...19:10 horas, se reunieron en el Auditorio de la Facultad de Ingeniería de Sistemas e Informática de la Universidad Nacional Mayor de San Marcos, el Jurado de Tesis conformado por los siguientes docentes:

Mg. Juan Carlos Gonzales Suárez (Presidente)
Mg. Zoraida Emperatriz Mamani Rodriguez (Miembro)
Mg. Anita Marlene Reyes Huamán (Miembro)
Dr. Glen Dario Rodriguez Rafael (Asesor)

Se inició la Sustentación invitando al candidato a Magíster ***Ronald Eduardo Ibarra Zapata****, para que realizara la exposición oral y pública de la tesis para optar el Grado Académico de Magíster en Ingeniería de Sistemas e Informática con mención en Ingeniería de Software, siendo la Tesis intitulada:*

"SOTESTER - Sistema de Recomendación de Técnicas de Testing de Software: Un Enfoque Colaborativo"

Concluida la exposición, los miembros del Jurado de Tesis procedieron a formular sus preguntas que fueron absueltas por el graduando; acto seguido se procedió a la evaluación correspondiente, habiendo obtenido la siguiente calificación:

.....16 DIECISEIS. BUENO.....

Por tanto el Presidente del Jurado, de acuerdo al Reglamento General de Estudios de Posgrado, otorga al Bachiller ***Ronald Eduardo Ibarra Zapata*** *el Grado Académico de Magíster en Ingeniería de Sistemas e Informática con mención en Ingeniería de Software.*

Siendo las...20:20..... horas, el Presidente del Jurado de Tesis da por concluido el acto académico de Sustentación de Tesis.

Mg. Juan Carlos Gonzales Suárez
(Presidente)

Mg. Zoraida Emperatriz Mamani Rodriguez
(Miembro)

Mg. Anita Marlene Reyes Huaman
(Miembro)

Dr. Glen Dario Rodriguez Rafael
(Asesor)

Dedico este trabajo a mi familia, en especial a Susana y Fernando.

AGRADECIMIENTOS

A Dios por la vida y salud que me da para cumplir mi misión en este mundo.

A mi familia por el apoyo durante mi formación y desarrollo profesional.

Al Dr. Glen Rodriguez por su apoyo académico incondicional durante mis estudios y realización de la tesis.

ÍNDICE GENERAL

RESUMEN x
ABSTRACT xii
CAPÍTULO 1. INTRODUCCIÓN 1
1.1. Situación problemática 3
1.2. Formulación del problema 4
1.2.1. Problema general 5
1.2.2. Problemas específicos 5
1.3. Justificación teórica 5
1.4. Justificación práctica 6
1.5. Objetivo General 6
1.6. Objetivos Específicos 6
1.7. Actividades 7
1.8. Taxonomía del tema según ACM 7
CAPÍTULO 2. MARCO TEORICO 8
2.1. Caracterización de técnicas y herramientas de testing de software 8
2.2. Estudios primarios para la evaluación de técnicas de testing 14
2.3. Framework para la realización de estudios primarios 14
2.3.1. Metodológico vs. Organizacional 18
2.3.2. Utilidad 18
2.3.3. Factores de evaluación 18
2.3.4. Instanciaciones 20
2.3.5. Mantenimiento de la base de conocimientos 21
2.4. Estudios secundarios para la evaluación de técnicas de testing 21
2.4.1. Problemas en el proceso de recomendación de técnicas y herramientas de testing de software 22
2.5. Sistemas de recomendación 23
2.5.1. Filtrado colaborativo 24
2.5.2. Basados en contenido 24
2.5.3. Trust 24
2.5.4. Aplicación de sistemas de recomendación en ingeniería de software 25
2.5.5. Evaluación de sistemas de recomendación 26
2.6. Enfoques para la toma de decisiones 27
2.7. Conclusiones del marco teórico 28
CAPÍTULO 3. METODOLOGÍA 30

3.1. Hipótesis de investigación 30
En base a los objetivos de la investigación, se plantean las siguientes hipótesis: 30
3.1.1. Hipótesis General 30
3.1.2. Hipótesis Específicas 30
3.2. Identificación y operacionalización de variables 30
3.3. Matriz de consistencia 33
3.4. Tipo y Diseño de Investigación 34
3.5. Unidad de análisis 34
3.6. Población de estudio 34
3.7. Tamaño de muestra 35
3.8. Técnicas de recolección de Datos 36
3.8.1. Observación directa 36
3.8.2. Cuestionario 36
3.8.3. Procedimiento 36
3.9. Análisis e interpretación de la información 37
3.10. Instrumentos de recolección de datos 38
3.11. Amenazas a la validez 39
CAPÍTULO 4. DISEÑO DE SOTESTER 41
4.1. Trabajos relacionados 41
4.2. Métodos de selección 42
4.2.1. Esquema de caracterización 43
4.2.2. Repositorio 44
4.3. Descripción del método de recomendación 45
4.3.1. Paso 1 - Caracterización del proyecto 46
4.3.2. Paso 2 - Elaborar ranking de técnicas 47
4.3.3. Paso 3 - Caracterizar técnicas instanciadas, ... 50
4.3.4. Paso 4 - Calificar recomendaciones 51
4.4. Construcción de la herramienta SOTESTER 51
4.4.1. Requisitos 52
4.4.2. Casos de uso 54
4.4.3. Clases de negocio 55
4.4.4. Estados de un proyecto 57
4.4.5. Vista de desarrollo 58
4.4.6. Vista física 59
4.4.7. Modelo físico de la base de datos 60
CAPÍTULO 5. RESULTADOS Y DISCUSION 70

5.1. Técnicas de testing 70
5.2. Proyectos históricos 71
5.3. Instanciación de técnicas 72
5.4. Proyectos Objetivo 73
5.5. Pruebas de Hipótesis 73
5.5.1. Hipótesis especifica 1 ($He1$) 74
5.5.2. Hipótesis especifica 2 ($He\mathbf{2}$) 76
5.5.3. Hipótesis general 78
CONCLUSIONES 80
TRABAJOS FUTUROS 82
REFERENCIAS 83
APENDICES 86
Apéndice 1 – FICHA DE RECOMENDACIÓN PARA EXPERTOS 86
Apéndice 2 – MATRICES DE ANÁLISIS DE RECOMENDACIONES HECHAS POR SOTESTER 88
Apéndice 4 – RECOMENDACIONES DE TÉCNICAS DE TESTING 110

Tablas

Tabla 1 - Esquema de clasificación presentado por Engström, Runeson, y Skoglund 9
Tabla 2 - Esquema de caracterización de Vegas y Basili 9
Tabla 3 - Esquema de caracterización presentado por Eldh 10
Tabla 4 - Esquema de caracterización presentado por Vos et al. 11
Tabla 5 - Esquema de caracterización MBT presentado por Dias y Travassos 12
Tabla 6 - Esquema de caracterización presentado por Cotroneo et al. 13
Tabla 7 - Frameworks y métodos para estudios primarios 16
Tabla 8 - Esquema de caracterización adoptado en SOTESTER 35
Tabla 8 - Comparativo con trabajos relacionados 42
Tabla 9 - Diferencias entre repositorios 45
Tabla 11 - Esquema de caracterización - Atributos de desempeño 51
Tabla 12 - Actores de casos de uso 54
Tabla 13 - Clases de negocio 55
Tabla 14 - Estados de un proyecto 57
Tabla 15 - Principales archivos de código fuente 58
Tabla 16 – Técnicas de testing de software 70
Tabla 17 – Perfil de profesionales que inicializaron el repositorio 71
Tabla 18 – Proyectos históricos 71
Tabla 19 – Instanciación de técnicas en proyectos históricos 72
Tabla 20 – Proyectos objetivo 73
Tabla 21. Coincidencias entre las recomendaciones de los expertos 74
Tabla 22 – Coincidencias entre recomendaciones de SOTESTER y expertos 76

Figuras

Figura 1 – Creando un cuerpo de evidencia. 15
Figura 2 – Repositorio. 21
Figura 3 – Estructura de matriz de análisis 39
Figura 4 - Repositorio propuesto en el presente trabajo 45
Figura 5 – Pasos del método SOTESTER 46
Figura 6 – Caracterización del proyecto 46
Figura 7 – Elaboración de ranking de técnicas 47
Figura 8 - Representación vectorial de un proyecto 48
Figura 9 - Cálculo de similitud relativa 48
Figura 10 – Caracterización de técnicas instanciadas 50
Figura 11 – Calificación de recomendaciones 51
Figura 12 – Casos de uso 55
Figura 13 – Diagrama de clases de negocio 56
Figura 14 – Diagrama de transición de estados de un proyecto. 58
Figura 15 – Diagrama de componentes - proyecto. 59
Figura 16 – Diagrama de despliegue 60
Figura 17 - Base de datos: vista de Proyecto 61
Figura 18 - Base de datos: Esquema de caracterización 62
Figura 19 - GUI: Registro de organización – parte 1 63
Figura 20 - GUI: Registro de organización – parte 2 64
Figura 21 - GUI: Crear proyecto 65
Figura 22 - GUI: Listado de proyectos 65
Figura 23 - GUI: Caracterización de proyecto 66
Figura 24 - GUI: Solicitud de recomendación 67
Figura 25 - GUI: Ranking de técnicas recomendadas 68
Figura 26 - GUI: Selección de técnicas instanciadas 68
Figura 27 - GUI: Caracterización de técnicas instanciadas 69
Figura 28 – Variables estadísticas para las pruebas de hipótesis 73
Figura 28 - Coincidencias en las técnicas recomendadas por los expertos 75
Figura 29 – Medidas descriptivas de las coincidencias en las técnicas recomendadas por los expertos 76
Figura 30 - Coincidencias en las técnicas recomendadas por SOTESTER y expertos 77
Figura 31 – Medidas descriptivas de las coincidencias en las técnicas recomendadas por SOTESTER y los expertos 78
Figura 32 – Diagrama de frecuencias de las coincidencias - Expertos vs. SOTESTER(System) 79
Figura 33 – Prueba de hipótesis: Mediana de las diferencias entre Expertos vs. Expertos y SOTESTER vs. Expertos 79

RESUMEN

Las pruebas de software constituyen un factor clave en un Proyecto de software; los costos de las pruebas son significativos en relación al costo de desarrollo. Por lo tanto, es esencial seleccionar las técnicas de testing más adecuadas para un proyecto dado para encontrar defectos al menor costo posible en los distintos niveles de pruebas. Sin embargo, en varios proyectos, los profesionales de testing no tienen un entendimiento profundo acerca de las técnicas de testing disponibles y adoptan las mismas técnicas que fueron usadas en proyectos previos, o alguna técnica disponible sin tener en consideración los atributos de cada técnica de testing.

Existen importantes trabajos de investigación orientados a la evaluación y selección de técnicas de testing de software, partiendo por esquemas de caracterización, y diversos enfoques para conducir estudios primarios orientados a la construcción de bases de conocimiento o repositorios; luego se han encontrado métodos y enfoques para la selección informada de técnicas de testing de software para un proyecto dado, en base a repositorios o bases de conocimiento de técnicas de testing de software; estos métodos están basadas en catálogos estáticos cuya adaptación en la industria podría ser lenta y costosa.

En el presente trabajo de investigación, se presenta un sistema de recomendación con enfoque colaborativo y basado en contenido que permite obtener recomendación de técnicas de testing de software basado en la caracterización del proyecto objetivo y la evaluación de técnicas de testing instanciadas en proyectos similares. Se ha demostrado que el método

propuesto SOTESTER, realiza recomendaciones buena calidad de manera similar a como las realiza un experto humano.

Palabras clave: “técnicas de testing de software”, “sistema de recomendación”, “repositorio colaborativo”, “razonamiento basado en contenido”, “K-vecino más cercano”

ABSTRACT

Software testing is a key factor on any software project; testing costs are significant in relation to development costs. Therefore, it is essential to select the most suitable testing techniques for a given project to find defects at the lower cost possible in the different testing levels. However, in several projects, testing practitioners do not have a deep understanding of the full array of techniques available, and they adopt the same techniques that were used in prior projects or any available technique without taking into consideration the attributes of each testing technique.

Currently, there are researches oriented to support selection of software testing techniques; from technique and project characterization schemas, approaches to lead primary studies and build knowledge bases about software testing techniques; to methods and approaches to informed selection of techniques based on repositories, nevertheless, they are based on static catalogues, whose adaptation to any niche software application may be slow and expensive.

As a theoretical basis, in this article we present an art state about the testing techniques evaluation and selection which contain a set of characterization schemas and a comparison between approaches for lead primary studies leading to build knowledge bases or repositories and in other way we present the existing methods and approaches about the informed selection of testing techniques for a given project based on repositories information.

In this work, we introduce a content-based recommender system that offers a ranking of software testing techniques based on a target project characterization and evaluation of testing techniques in similar projects. The

repository of projects and techniques was completed through the collaborative effort of a community of practitioners. It has demonstrated that purposed method SOTESTER makes good quality recommendations in similar way as a human expert group.

Keywords: “software testing techniques”, “recommender system”, “collaborative repository”, “content-based reasoning”, “K-Nearest Neighbors”

CAPÍTULO 1. INTRODUCCIÓN

"Las pruebas de software son procesos diseñados para asegurar que el código del computador haga aquello para lo que fue diseñado. El principal propósito de las pruebas puede ser: aseguramiento de calidad, estimación de fiabilidad, validación o verificación" (Khan, 2010, p. 11). Además, las pruebas de software están presentes en todo el ciclo de vida y se aplican diferentes técnicas en cada nivel de pruebas: pruebas unitarias, de integración, del sistema y de aceptación. Así mismo, las pruebas de software son costosas para la industria y siempre son limitadas por tiempo y esfuerzo (Eldh, Hansson, Punnekkat, Pettersson, y Sundmark, 2006).

"Las técnicas de testing de software, están basadas en un amalgama de métodos extraídos de la teoría de grafos, lenguajes de programación, evaluación de fiabilidad, teoría de prueba confiable, etc." (Luo, 2001, p. 5), lo cual representa cierta complejidad y necesidad de esfuerzo para su aplicación; por lo que considerando además las consecuencias de los fallos y sus costos asociados, es muy importante la selección de los métodos de pruebas más eficientes y efectivos.

Aunque hay muchas técnicas de pruebas, no hay guías científicas para su selección apropiada en los diferentes dominios y contextos. Hay muchas técnicas, lo cual hace complejo el proceso de selección porque los profesionales quieren saber cuáles de las técnicas detectarán los defectos de su interés en la mayoría de los programas que ellos planean probar (Farooq y Quadri, 2013). Una consecuencia es que en la práctica se utilizan técnicas de pruebas de software seleccionándolas de manera empírica según la experiencia de cada profesional, lo cual representa un problema, ya que los

empleados se movilizan a menudo entre una organización y otra, por lo que los criterios y puntos de vista para la selección de técnicas de pruebas serían subjetivos y poco sistemáticos.

Existen varios esfuerzos en el mundo académico y en la industria para contar con marcos de trabajo que permitan conducir estudios primarios para caracterizar las técnicas de pruebas de software de manera uniforme para generar bases de conocimiento y evidencia incremental útil para la realización de estudios secundarios de manera cotidiana en la industria del software con la finalidad de evaluar las técnicas y herramientas de pruebas de software en contextos particulares de modo que se garantice principalmente efectividad y eficiencia en el proceso de pruebas.

Por otro lado, existen varios métodos y estrategias que se proponen para apoyar el proceso de selección de una o más técnicas de pruebas de software para un contexto dado generalmente por las características del proyecto y sistema a probar.

El presente trabajo de investigación propone un método de recomendación de técnicas de testing de software basado en un enfoque colaborativo, para cuya implementación, se construye una aplicación web.

En el presente capítulo, se expone una introducción al tema de investigación además de la situación, descripción y formulación del problema de investigación, así como la justificación y objetivo de la investigación.

En el Capítulo 2, se presenta el marco teórico elaborado mediante revisión de literatura, teniendo en cuenta, la caracterización de técnicas de testing de software, estudios primarios y secundarios para la evaluación de técnicas de testing, sistemas de recomendación y finalmente, los enfoques para la toma de decisiones.

En el Capítulo 3, se presentan los aspectos metodológicos de la investigación, tales como el planteamiento de la hipótesis, las variables de investigación y su operacionalización; así como aspectos relacionados a los procedimientos e instrumentos para la recolección de datos.

En el Capítulo 4, se presenta el diseño del método propuesto SOTESTER, haciendo comparaciones con los trabajos previos relacionados para cada componente del método; también se presenta una descripción acerca de la construcción de la aplicación web que implementa el método.

En el Capítulo 5, se muestran los resultados y discusiones, luego de haber utilizado el método SOTESTER para obtener recomendaciones, realizando una comparación con las recomendaciones realizadas por expertos humanos.

Hacia el final del presente informe, se presentan las conclusiones de la investigación, así como algunas sugerencias a manera de trabajos futuros; seguidas de las referencias bibliográficas y anexos que en general muestran evidencias que podrían brindar mayores detalles acerca de los datos recopilados en la presente investigación, así como facilitar su replicación.

1.1. Situación problemática

Respecto a la necesidad de seleccionar las técnicas de testing más adecuadas en distintos escenarios en la industria, Vegas y Basili (2005) afirman:

> Aunque la cuestión de cuáles son las técnicas más adecuadas para desarrollar el banco de casos de prueba para probar un Sistema dado, aparentemente posee enormes dificultades, esta es sin embargo una cuestión con la que lidian los testers cada vez que tienen que probar un sistema. ¿Y cómo se responde actualmente a esta cuestión?: Ni sistemáticamente ni con directrices bien definidas. En efecto, de todas las técnicas existentes, algunas nunca son consideradas para usarse y otras son usadas una y otra vez en diferentes proyectos sin siquiera ser examinadas después de su uso sean o no realmente útiles. (p. 438)

Así mismo, Vegas y Basili (2005) indican que las decisiones tomadas por los desarrolladores no son tomadas tan al azar en la medida de su conocimiento de la técnica. También indican que hay dos razones principales por las que los desarrolladores no toman buenas decisiones: la primera es que la información disponible sobre las técnicas está normalmente distribuida a través de diferentes fuentes de información (libros, artículos e incluso la gente). Esto quiere decir que los desarrolladores no tienen una idea general

de que técnicas están disponibles y de toda la información de interés acerca de cada técnica de *testing*. Y la segunda es que no tienen acceso a información pragmática concerniente a cada técnica de prueba a menos que ellos la hayan usado antes. Los desarrolladores no tienden a compartir con otros el conocimiento que adquieren al usar técnicas de pruebas; deduciendo que los desarrolladores pierden la oportunidad de aprender a partir de las experiencias de otros.

En concordancia con Vegas y Basili (2005), Eldh et al. (2006) expresan que los profesionales de *testing* tienen poca o ninguna información acerca de las técnicas disponibles, su utilidad y generalmente cuan adecuadas son para sus proyectos sobre lo cual basar su decisión respecto a qué técnica utilizar.

Actualmente a nivel de investigación, se han propuesto varios marcos de trabajo, protocolos y listas de chequeo orientados a soportar la selección de técnicas y herramientas de *testing*, estos caracterizan tanto el software a probar, como la técnica y/o herramienta de prueba a utilizar. De este modo tal como lo indican Vos, Marín, Escalona y Marchetto (2012), se pretende que la información a partir de estudios primarios pueda convertirse en una base de evidencia para estudios secundarios que soporten la selección de técnicas y herramientas de *testing*. Sin embargo, existen limitaciones de tiempo y recursos para realizar los estudios primarios que hasta el momento han sido realizados mediante casos de estudio y experimentos en algunas ocasiones en la industria y en otras en ambientes académicos.

1.2. Formulación del problema

Los métodos de selección o recomendación de técnicas de testing de software existentes hasta el momento, se basan en catálogos estáticos elaborados generalmente mediante revisión de literatura, y el mantenimiento de su base de conocimientos depende de varios actores (investigadores, profesionales, encargado de registrar la información), dificultándose su aplicación en la industria debido al tiempo y esfuerzo que esto implica.

Como una solución al problema mencionado, como parte de la presente investigación, se propone e implementa un método de recomendación de técnicas de testing denominado SOTESTER, el cual está basado en catálogo dinámico alimentado de manera colaborativa por profesionales de la industria de software, por lo que es importante comprobar si las recomendaciones que realiza el método son similares a las de un conjunto de verdad, en este caso las recomendaciones hechas por expertos humanos. Por ello, se plantea la siguiente pregunta de investigación:

1.2.1. Problema general

¿Cuál es la calidad de las recomendaciones de técnicas de testing de software realizadas mediante el Método propuesto SOTESTER respecto a las recomendaciones de técnicas de testing de software realizadas por expertos?

1.2.2. Problemas específicos

- ¿En qué medida coinciden los expertos respecto a las técnicas de testing de software que recomiendan?
- ¿En qué medida coincide el método propuesto SOTESTER con los expertos respecto a las técnicas de testing de software que recomiendan?

1.3. Justificación teórica

La presente investigación busca aportar al conocimiento existente acerca de la selección informada de técnicas de testing de software en el campo de la ingeniería de software, así como la aplicabilidad de diversos enfoques y técnicas de sistemas de información, tales como métodos de recomendación, técnicas para determinar similitud y filtrado colaborativo. Los resultados de la presente investigación estarían incorporándose como conocimiento en el campo de ingeniería de software dado que se estaría demostrando que es posible seleccionar técnicas de testing de software mediante un sistema de recomendación con un enfoque colaborativo.

1.4. Justificación práctica

El presente trabajo responde a la necesidad de la industria de mejorar la calidad del testing en los proyectos de software mediante la selección informada de técnicas de testing.

En la industria, las micro, pequeñas y medianas organizaciones que producen software, al contar con un método y herramienta de recomendación, tendrían la oportunidad de:

- Permitirse hacer una adecuada selección de técnicas de testing para sus proyectos sin necesariamente tener que contar con expertos en técnicas de testing, lo cual permitirá optimizar esfuerzo y costos.
- Disminuir sus costos de no calidad que pueden producirse por no seleccionar las técnicas de testing más adecuadas a sus proyectos.

1.5. Objetivo General

Medir la calidad de las recomendaciones de técnicas de testing de software realizadas por el método propuesto SOTESTER respecto a las recomendaciones de técnicas de testing de software realizadas por expertos.

1.6. Objetivos Específicos

- Medir la coincidencia de los expertos respecto a las técnicas de testing de software que recomiendan.

- Medir la coincidencia del método propuesto SOTESTER con los expertos respecto a las técnicas de testing de software que recomiendan.

1.7. Actividades

Como parte de la investigación, teniendo en cuenta que se trata de una propuesta para resolver un problema práctico, se deben realizar las siguientes actividades, las cuales permitirán cumplir con los objetivos de la investigación:

- Definir el método de recomendación de técnicas de testing de software mediante un enfoque colaborativo y catálogo dinámico, que permita seleccionar técnicas de testing de software para un proyecto determinado en la industria, aprovechando la experiencia de uso de técnicas de testing en otros proyectos.
- Diseñar y construir una herramienta web colaborativa y escalable que implemente el método de recomendación y permita el mantenimiento de un repositorio o base de conocimiento de proyectos e instanciaciones de técnicas de testing de software.
- Validar las recomendaciones ofrecidas por el método de recomendación propuesto.

1.8. Taxonomía del tema según ACM

Software and its engineering - Software verification and validation

- Software defect analysis - Software testing and debugging

Information systems - Information retrieval

- Retrieval tasks and goals - Recommender systems

CAPÍTULO 2. MARCO TEORICO

Se realiza una revisión de literatura orientada a establecer los trabajos previos en materia de selección de técnicas de testing de software. Para lo cual se considera la caracterización de las técnicas de testing de software, seguido por los enfoques para la realización de estudios primarios, los trabajos existentes para la selección de técnicas de testing mediante estudios secundarios, así como una breve revisión de sistemas de recomendación y métodos para la toma de decisiones.

2.1. Caracterización de técnicas y herramientas de testing de software

Existen varios trabajos de investigación (Lawanna, 2014), (Arora y Sinha, 2012), (Brosse, Vos, y Condori, 2014), (Cotroneo, Pietrantuono, y Russo, 2013), (Muthusamy y Seetharaman, 2014) y (Vos et al., 2012) que hacen referencia al estudio y evaluación de distintas técnicas de *testing* de software, sin embargo se dificulta la comparación de los resultados de dichos estudios debido a que se usan distintos esquemas de clasificación y/o caracterización. A continuación, se realizará un análisis a alto nivel de los esquemas de clasificación y/o caracterización encontrados en diversos trabajos de investigación.0

En un trabajo de investigación, Engström, Runeson, y Skoglund, utilizan el esquema de caracterización mostrado en la Tabla 1, el cual consta de 7 atributos o características de carácter cualitativo, es de propósito general, orientado a la clasificación y es de gran utilidad para una clasificación preliminar de las técnicas de testing existentes, puesto que ha sido

instanciado de manera exitosa para la clasificación de un total de 28 técnicas en el dominio de pruebas de regresión como parte de un trabajo de revisión de literatura. (Engström, Runeson, y Skoglund, 2010)

Tabla 1 **- Esquema de clasificación presentado por Engström, Runeson, y Skoglund**

Elemento	Atributo	Valor
Aplicabilidad	Tipo de lenguaje	Procedural, OO, Independiente
	Tipo de software	Basado en componentes, Manejador de base de datos
Método	Entrada	Código fuente, Código intermedio para máquinas virtuales, Código de máquina, Entradas en ciertos formatos
	Enfoque	Control de flujo, Firewall, Slicing, Basado en dependencia
	Granularidad	Sentencia, Función, Clase, Modulo, Componente
Propiedades	Habilidad de detección	Segura
	Reducción de costos	Minimización

Fuente. Datos tomados de (Engström et al., 2010)

Vegas y Basili, proponen el esquema de caracterización mostrado en la Tabla 2, el cual consta de 32 atributos entre cualitativos y cuantitativos, organizados además en 2 niveles: operacional e histórico. Este esquema además de haber sido elaborado de manera iterativa contemplando la perspectiva por un lado de quienes producen información acerca de las técnicas de testing, y por otro lado de quienes utilizan el esquema y su información para seleccionar técnicas de testing; ha sido validado exitosamente de manera experimental usándose para construir un catálogo de técnicas de testing (Vegas y Basili, 2005).

Tabla 2 **- Esquema de caracterización de Vegas y Basili**

Nivel	Elemento	Atributo	Descripción
Operacional	Agente	Conocimiento	Conocimiento requerido para poder aplicar la técnica
		Experiencia	Experiencia requerida para poder aplicar la técnica
	Herramientas	Identificador	Nombre de la herramienta y del fabricante
		Automatización	Parte de la técnica automatizada por la herramienta
		Costo	Costo de la herramienta compra y mantenimiento
		Entorno	Plataforma (software y hardware) y lenguaje de programación que opera la herramienta
	Técnica	Soporte	El soporte prestado por el fabricante de la herramienta
		Comprensibilidad	Es o no la técnica fácil de entender

		Costo de aplicación	Cuánto esfuerzo se necesita para aplicar la técnica
		Entradas	Entradas necesarias para aplicar la técnica
		Criterio de adecuación	Generación de casos de prueba y la regla de parada
		Costo para la prueba de datos	Costo para identificar los datos de prueba
		Dependencias	Relaciones de una técnica con otra
		Repetibilidad	Si dos personas generan los mismos casos de prueba
	Casos de prueba	Fuentes de información	Donde encontrar información acerca de la técnica
		Completitud	La cobertura proporcionada por el conjunto de casos
		Precisión	Cuantos casos de pruebas repetidas genera la técnica
		Eficacia	Que capacidad del conjunto de casos debe tener para detectar defectos
		Tipo de defecto	Tipos de defectos detectados en el sistema
		Número de casos generados	Número de casos generados por el tamaño de la unidad de software
	Objeto	Elemento	Elementos del sistema sobre el que actúa la prueba
		Aspecto	Funcionalidad del sistema a ensayar
		Tipo de software	Tipo de software que puede ser probado usando la técnica
		Lenguaje de programación	Lenguaje de programación que se puede utilizar
		Método de desarrollo	Método de desarrollo o ciclo de vida a la que está vinculado
		Tamaño	Tamaño que el software debe tener para poder utilizar la técnica
	Proyecto	Proyectos de referencia	Proyectos anteriores en los que la técnica se ha utilizado
		Herramientas usadas	Las herramientas utilizadas en proyectos anteriores
Histórico		Personal	El personal que trabajaron en proyectos anteriores
	Satisfacción	Opinión	Opinión general sobre la técnica después de haber usado
		Beneficios	Beneficios del uso de la técnica
		Problemas	Problemas con el uso de la técnica

Fuente. Datos tomados de (Vegas y Basili, 2005)

En la Tabla 3, se muestra el esquema de caracterización presentado por Eldh et al., el cual consta de 17 atributos entre cualitativos y cuantitativos orientados a medir la eficiencia, eficacia y aplicabilidad de la técnica. Este esquema ha sido utilizado como parte de un framework experimental para comparación de técnicas de testing (Eldh et al., 2006).

Tabla 3 **- Esquema de caracterización presentado por Eldh**

Medida de eficiencia
1. Tiempo actual para cada fase
2. Tiempo para detector defectos y/o fallas, tiempo para detector el tipo de defecto.

3. Cuánto tiempo toma encontrar el primer defecto o falla
4. Opinión personal de los sujetos sobre cada tarea en el proceso (fácil, difícil, posee problemas secundarios)
5. Tiempo para crear manualmente los casos de prueba (para uno, el primero y variantes)
6. Cuantos casos de prueba únicos e instancias de casos de prueba se crean.
Medida de eficacia
7. Número absoluto de cuantos defectos no inyectados, fueron encontrados comparados con los defectos inyectados.
8. Cantidad de defectos aislados por tipo y severidad del total de defectos encontrados.
9. Estimación de cobertura
Aplicabilidad de la técnica
10. En qué fase son encontrados(distribuidos) los defectos en el tiempo
11. Juicio personal del sujeto de cada tarea en el proceso (Fácil, difícil, plantea problemas secundarios, etc.)
12. Facilidad de aprendizaje de la técnica (fácil, difícil, posee problemas secundarios, etc.)
13. Niveles (código, componente, integración, subsistema, sistema) donde es posible usar la técnica.
14. Generalidad de la técnica
a. Contexto (Sistema operativo, hardware, dominio, etc.)
b. Lenguaje y restricciones (C, C++, Java, y versión/compilador)
15. Número de variantes de la técnica dentro de cada alcance
16. Evaluación de aplicabilidad de automatización de la técnica, lo cual es una evaluación cualitativa. La medida se encuentra en el rango a partir de (mala, lenta, ineficaz) en una escala flotante hasta (buena, rápida y eficaz)
17. Evaluación del proceso completo.

Fuente. Datos tomados de (Eldh et al., 2006)

Vos et al., presentan un esquema de caracterización de carácter general, el cual se muestra en la Tabla 4. Este esquema (Vos et al., 2012) se basa en el propuesto por Vegas y Basili (Vegas y Basili, 2005) y consta de 22 atributos cualitativos y cuantitativos; y ha sido utilizado exitosamente en ambientes de la industria como parte de un framework metodológico para la evaluación de técnicas de testing de software. Los atributos del esquema coberturan los siguientes aspectos: prerrequisitos para el uso de la técnica, resultados de instanciar la técnica, operación de la técnica, obtención de la herramienta para el uso de la técnica.

Tabla 4 **- Esquema de caracterización presentado por Vos et al.**

Elemento	Atributo
Prerequisitos	Estática o dinámica
	Tipo de software
	Fase de ciclo de vida
	Entorno
	Escalabilidad
	Entradas
	Conocimiento
	Experiencia
Resultados	Salidas
	Completitud
	Eficacia
	Tipos de defectos

	Tamaño del banco de pruebas
Operación	Interacción
	Guía al usuario
	Fuentes de información
	Tareas de aplicabilidad
	Comprensibilidad
	Satisfacción subjetiva
	Esfuerzo
	Madurez
Obtención de la herramienta	Obtención de la herramienta

Fuente. Datos tomados de (Vos et al., 2012)

Dias y Travassos presentan un esquema orientado a caracterizar técnicas MBT (Model Based Testing), el cual se muestra en la *Tabla 5*. Este esquema también se basa en el esquema propuesto por Vegas y Basili (2005), consta de 25 atributos cualitativos adecuados para caracterizar técnicas MBT, estos atributos no solo caracterizan la técnica si no también el proyecto en el cual se instancia. Este esquema ha sido utilizado como parte de un método de selección combinada de técnicas de testing MBT (Dias y Travassos, 2014).

Tabla 5 - **Esquema de caracterización MBT presentado por Dias y Travassos**

Código	Atributo
CA01	Modelo usado para representar el comportamiento/estructura del software a probar.
CA02	Criterio usado para medir la cobertura de las pruebas
CA03	Criterio usado para generar automáticamente los casos de prueba
CA04	Entradas requeridas para empezar a usar la técnica MBT
CA05	Limitaciones/restricciones para usar una técnica MBT
CA06	Tipo de testing en que la técnica se puede aplicar.
CA07	Nivel de testing en el cual la técnica es aplicada.
CA08	Formato de las salidas generadas por una técnica MBT
CA09	Características de calidad del software que una técnica MBT puede evaluar.
CA10	Existencia (o no) de una herramienta que soporte la técnica
CA11	Costo requerido para usar la herramienta de soporte.
CA12	Plataforma de ejecución para usar la herramienta de soporte.
CA13	Plataforma de ejecución del software bajo el cual la técnica MBT se puede ejecutar.
CA14	Paradigma de desarrollo de software bajo el cual la técnica MBT puede probar el software.
CA15	Lenguaje de programación con el cual la técnica MBT puede probar el software.
CA16	Necesidad de desarrollo o modelos intermedios durante el proceso de generación de pruebas.
CA17	Tecnología usada para modelar las pruebas generadas.
CA18	Necesidad de herramientas externas durante la generación de pruebas.
CA19	Resultados históricos del uso de una técnica MBT en proyectos de software previos
CA20	Proporción de pasos automatizados respecto al total de pasos en el proceso de generación de pruebas.
CA21	Indicación de evaluación experimental de la eficiencia y escalabilidad de la técnica MBT.

CA22	Nivel de complejidad para ejecutar los pasos no automatizados durante el proceso de generación de pruebas.
CA23	Existencia de un mecanismo de trazabilidad entre los modelos de software y las pruebas generadas
CA24	Habilidades/Conocimiento requerido para usar la técnica MBT.
CA25	Existencia de modelo de verificación para evaluar los modelos antes de la generación de pruebas.

Fuente. Datos tomados de (Dias y Travassos, 2014)

Cotroneo et al. presentan un esquema de caracterización con un total de 17 atributos en forma de interrogantes que al parecer involucran la recopilación de ciertas métricas cuantitativas y cualitativas. Sus atributos se organizan en 3 niveles: Habilidad para la detección de defectos, costo de detección de defectos y actitud hacia el tipo de defectos, para efectos del presente análisis, en la *Tabla 6*, se presentan estos 17 atributos más 3 adicionales que se usaron en el mismo trabajo para caracterizar el software a probar. Este esquema ha sido utilizado experimentalmente para validar un método de selección de técnicas de testing basado en el tipo de defectos y métricas del software (Cotroneo et al., 2013).

Tabla 6 - **Esquema de caracterización presentado por Cotroneo et al.**

Técnica de testing	**Capacidad de detección de defectos**
	1. ¿Cuál de las técnicas comparadas detecta el mayor porcentaje de defectos respecto a los presentes inicialmente? 2. ¿El porcentaje de fallos detectados depende de la cantidad inicial de fallos en el programa? 3. ¿El porcentaje de defectos detectados depende del tipo de software elegido? **Costo de detección de defectos** 1. ¿Cuál de las técnicas detecta el mayor número de defectos en el mismo tiempo de prueba (tasa: El mayor # de defectos/h) asumiendo el esfuerzo personal como medida de costo (hora persona). 2. El ratio #defectos/h (tasa de detección) depende de la cantidad inicial de defectos? 3. ¿La tasa de detección depende del tipo de software elegido? **Actitud hacia el tipo de fallos** 1. Dado un tipo de defecto, ¿Cuál de las técnicas disponibles detecta el mayor porcentaje de defectos de tal tipo (son los defectos de un tipo as fácilmente detectable por una técnica en particular que con otra)? 2. ¿Es una técnica dada más propensa a detector algún tipo de defecto que otras?
Software a probar	Tamaño del programa Complejidad del programa Ciencia del software Métricas de Halstead

Fuente. Datos tomados de (Cotroneo et al., 2013)

2.2. Estudios primarios para la evaluación de técnicas de testing

En varios trabajos de investigación (Cotroneo et al., 2013), (Vegas y Basili, 2005), (Vos et al., 2012), se menciona la necesidad de realizar estudios primarios que proporcionen una base de evidencia y/o conocimiento como soporte para la evaluación y selección de técnicas y/o herramientas de testing que realizan los profesionales de la industria e investigadores. El proceso de selección podría realizarse mediante estudios secundarios, consiguiendo de este modo tomar decisiones de manera informada, lo cual reduciría los riesgos asociados a una inadecuada selección.

En la literatura científica, se suele encontrar muchos estudios primarios en los cuales comparan el desempeño de técnicas de testing, o analizan el desempeño de cada una de ellas de manera independiente; generalmente mediante experimentos y casos de estudio en ambientes académicos y de la industria, siguiendo diversos enfoques, protocolos, enfoques y marcos de trabajo.

Para la realización de estudios primarios en la industria, se han identificado varias barreras como: el consumo de tiempo, costos , resistencia de las empresas a participar en casos de estudio; sin embargo estas barreras deben superarse para lograr acceder a las ventajas que supone la toma de decisiones informada como por ejemplo: contar con información estructurada que permita realizar comparaciones entre técnicas y/o herramientas de testing, minimizar los riesgos de una selección inadecuada de técnicas de testing, tomar decisiones en base a evidencia y experiencia objetiva, dar lugar al desarrollo de herramientas que den soporte a la toma de decisiones, entre otros (Vos et al., 2012).

2.3. Framework para la realización de estudios primarios

La necesidad de contar con frameworks, esquemas, entre otros artefactos para dar soporte al proceso de evaluación de técnicas de software, ha sido

propuesta en diferentes trabajos de investigación, así Vegas y Basili dicen que utilizando un esquema de caracterización de técnicas y/o herramientas de testing de software, se puede construir un repositorio que contenga la descripción de cada técnica de interés y describa todas las técnicas de acuerdo al mismo patrón de modo que pueda tomarse una decisión sobre la conveniencia de usar una técnica sin necesidad de tener conocimiento procedimental de la técnica (Vegas y Basili, 2005). Por otro lado y con el mismo objetivo, en dos trabajos de investigación adicionales (Vos, Mar, Panach, Baars, y Ayala, 2011), (Vos et al., 2012), se refiere la necesidad de contar con un framework que debe ser instanciado para realizar estudios primarios con fines de evaluar una técnica o herramienta de testing con casos de estudio diseñados en base a un mismo framework, tal como se muestra en la Figura 1, para lo cual a su vez es necesario el uso de un esquema de caracterización como los indicados en la Tabla 1 y Tabla 2.

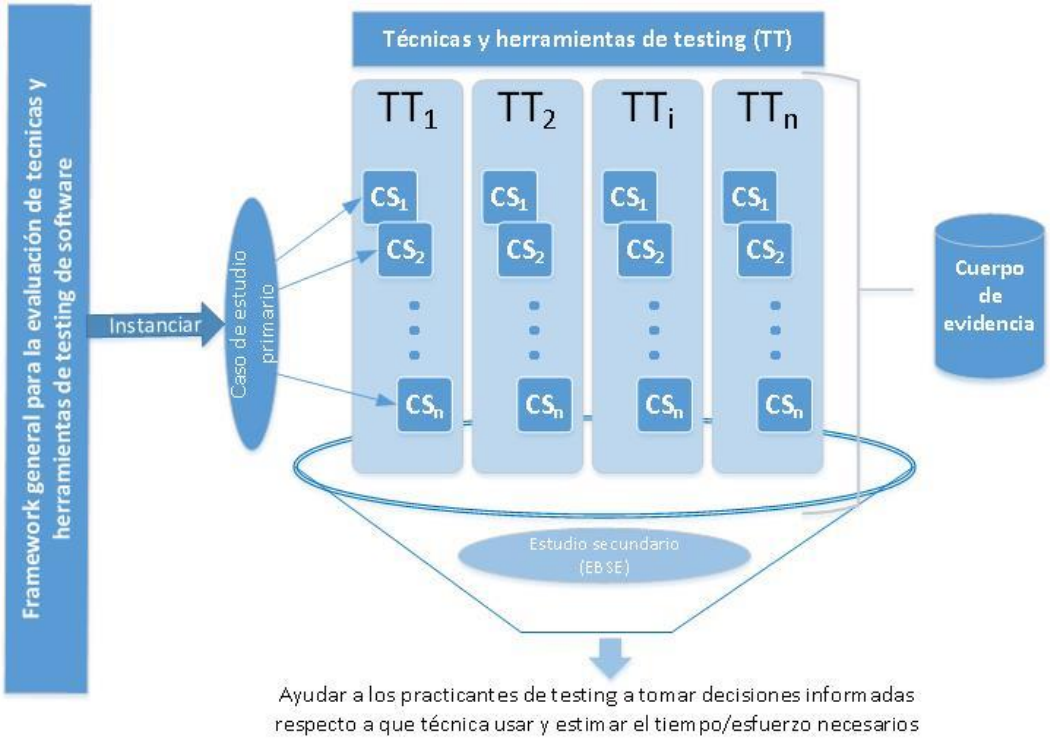

Figura 1 – **Creando un cuerpo de evidencia.**
Fuente (Vos et al., 2012)

En la Tabla 7, se presenta una comparación de cuatro frameworks para estudios primarios encontrados en múltiples trabajos de investigación (Cotroneo et al., 2013), (Eldh et al., 2006), (Vos et al., 2011) (Vos et al., 2012).

Tabla 7 - **Frameworks y métodos para estudios primarios**

Framework / Método	(Eldh et al., 2006)	(Vos et al., 2011)	(Vos et al., 2012)	(Cotroneo et al., 2013)
Tipo	Metodológico	Metodológico		Organizacional
Puede usarse para	Diseño de experimentos controlados	Diseño de casos de estudio		Diseño de experimentos controlados
Alcance	Técnicas	Técnicas y Herramientas		Técnicas
Limitado a	Inyección de defectos	Es genérico, no tiene limitaciones		Inyección de defectos
Evaluación	Simple	Simple		Múltiple
Factores de evaluación	Eficacia Eficiencia Aplicabilidad	Eficacia Eficiencia	Eficacia Eficiencia Satisfacción subjetiva	Habilidad para detectar defectos Costo detección defectos Actitud hacia tipo de defecto
Base de criterios de medición	No indicada	ISO 9241-11: 1998		ODC (Chillarege et al., 1992)
Instanciaciones	Académicas	Académicas y en la industria		Académicas
Técnicas para las que fue instanciado	Random, Fast Anti-Random, Equivalence classes, Boundary value, heuristic method	Search based structural testing, Search Based Functional Testing, Web testing: model-based testing, coverage-based testing, black-box testing and state-based testing		Functional Statistical Stress Robustness
Alcance : técnicas	Test case design	Test case design		Cualquier técnica de testing

Fuente. Elaboración propia en base a datos tomados de (Cotroneo et al., 2013), (Eldh et al., 2006), (Vos et al., 2011) y (Vos et al., 2012)

El método propuesto por Cotroneo et al. comprende dos grandes pasos: el primero, consiste en construir un conjunto de modelos predictivos para caracterizar una aplicación de software desde la perspectiva del tester quien es el interesado en saber cuántos defectos tiene la aplicación y de qué tipo usando ODC(Orthogonal Defect Classification); el paso siguiente define un procedimiento basado en experimentos controlados para caracterizar las técnicas de testing en términos de tipos de defectos ODC que son propensas a detectar; una vez elegido el conjunto de técnicas de interés, la caracterización consiste en evaluar su performance respecto a tipos de defectos ODC que potencialmente pueden afectar al sistema, costo de detección y eficacia en la detección. De este modo podría relacionarse el sistema a probar con las técnicas de testing mediante el tipo de defectos ODC

que el sistema es propenso a contener y que una o más técnicas tiende a detectar (Cotroneo et al., 2013).

En el trabajo de investigación de Eldh et al. se propone un framework para comparar la eficiencia, eficacia y aplicabilidad de técnicas de testing de software, orientado al diseño de experimentos, el cual consiste en los siguientes pasos: (1) preparar muestras de código con defectos conocidos, (2) seleccionar una técnica de testing, (3) realizar el experimento aplicando la técnica y recolectar datos, finalmente (4) analizar los datos, comparar las técnicas y evaluar resultados. El experimento debe repetirse con distintas técnicas y con nuevas muestras de código (con nuevos defectos inyectados) hasta tener suficientes datos para realizar sugerencias (Eldh et al., 2006).

En el trabajo de investigación de Vos et al. (2011) se propone un framework metodológico orientado al diseño de casos de estudio cuyos criterios de evaluación son la eficiencia y la eficacia; y que consta de procedimientos e indicaciones exhaustivas organizados en nueve partes: (1) Objetivo - ¿Que se espera conseguir?, (2) Casos o tratamiento - ¿Que se estudia o evalúa?, (3) Sujetos - ¿Quiénes aplican las técnicas/herramientas?, (4)Objetos - ¿Cuáles son los proyectos piloto?, (5) Proposición o hipótesis concreta - ¿Qué se quiere conocer?, (6) Variables y métricas - ¿Qué datos recolectar?, (7)Protocolo - ¿Cómo ejecutar el estudio y recolectar los datos?, (8)Análisis de datos - ¿Cómo interpretar los hallazgos?, (9)Amenazas a la validez de los estudios realizados (Vos et al., 2011).

Vos et al. (2012) proponen un framework metodológico (Vos et al., 2012) que mejora el framework propuesto anteriormente (Vos et al., 2011), agregando la satisfacción subjetiva como variable de evaluación de técnicas y herramientas de testing, y agregándose un esquema de caracterización basado en el esquema propuesto por Vegas en su tesis doctoral (Vegas, 2002).

A continuación, se resaltan algunos aspectos en que se parecen y/o diferencian los frameworks arriba mencionados:

2.3.1. Metodológico vs. Organizacional

En tres trabajos de investigación (Eldh et al., 2006), (Vos et al., 2011), (Vos et al., 2012) se presentan frameworks metodológicos, mientras que en uno (Cotroneo et al., 2013) se presenta un framework organizacional. Un framework organizacional, generalmente está enfocado en aportar en el proceso de evaluación de técnicas de testing mediante esquemas de caracterización, líneas guías, precauciones a tener en cuenta para que los estudios sean diseñados cuidadosamente minimizando los factores de confusión; a diferencia del metodológico que propone el: cómo evaluar, cómo definir las preguntas de investigación, las variables que deben medirse, y que amenazas a la validez especificas pueden presentarse.

2.3.2. Utilidad

Un framework puede utilizarse para realizar estudios experimentales tal como se propone en dos trabajos de investigación (Cotroneo et al., 2013), (Eldh et al., 2006); o para la conducción de casos de estudio como se propone en otros dos trabajos de investigación (Vos et al., 2011), (Vos et al., 2012). Es importante resaltar que en el caso de frameworks orientados al diseño de experimentos para la evaluación de técnicas de testing, generalmente requieren hacer uso de la inyección de defectos, mientras que en los de casos de estudio, esto es opcional.

2.3.3. Factores de evaluación

Todos los frameworks presentan coincidencias a alto nivel en cuanto a los factores de evaluación de las técnicas y/o herramientas de testing de software, sin embargo, difieren en cuanto a los atributos o indicadores necesarios para su medición.

En dos de los frameworks estudiados (Vos et al., 2011), (Vos et al., 2012), la eficiencia es determinada a partir de:

- Tiempo para el aprendizaje de la técnica
- Tiempo para diseñar los casos de prueba

- Tiempo para instalar la infraestructura de pruebas necesaria para la técnica
- Tiempo para probar y observar fallas
- Tiempo para identificar tipos de defectos y causas para cada falla observada.

Mientras que en otro de los frameworks (Eldh et al., 2006), la eficiencia es determinada a partir de:

- Tiempo (planificación, implementación, ejecución)
- Tiempo para detectar defectos e identificar su tipo, tiempo para encontrar el primer defecto o la primera falla, juicio de cada sujeto participante en las tareas de prueba (fácil, difícil, etc.),
- Tiempo para crear casos de prueba manualmente, cantidad de instancias únicas de los casos de prueba son creadas (número por caso de prueba/número de variantes).

En dos frameworks (Vos et al., 2011), (Vos et al., 2012), la eficacia se determina a partir de:

- Número de casos de prueba diseñados o generados
- Número de casos de prueba generados inválidos
- Número de casos de prueba generados repetidos, numero de fallas observadas, numero de defectos encontrados
- Numero de falsos positivos,
- Numero de falsos negativos
- Tipo y causa de los defectos encontrados
- Estimación de cobertura

Mientras que en uno de los frameworks estudiados (Eldh et al., 2006) la eficacia se determina a partir de:

- Numero de defectos encontrados comparado con los inyectados
- Defectos encontrados por cada nivel de severidad del defecto
- Estimación de cobertura de flujo de datos y control de flujo.

Es importante mencionar la similitud entre satisfacción subjetiva y aplicabilidad que presentan como factores de evaluación dos de los framewoks estudiados. El primero (Vos et al., 2012) considera como criterio

de evaluación la satisfacción subjetiva a partir de indicadores como: percepción de complejidad, intención de uso futuro, facilidad de uso, necesidad de soporte adicional, percepción de inconsistencias, facilidad de aprendizaje, etc., utilizando para su medición un cuestionario denominado System Usability Score (SUS), diagramas de Ven y nube de palabras para agrupar reacciones de los sujetos, así como reacciones faciales emocionales durante las entrevistas opiniones subjetivas acerca de las técnicas y / o herramientas evaluadas; el segundo framework (Eldh et al., 2006) a su vez contempla como criterio de evaluación, la aplicabilidad a partir de indicadores como: fase en la que se encuentran los errores, facilidad de aprendizaje, niveles donde es posible usar la técnica, evaluación del proceso completo, generalidad de la técnica (SO, hardware, dominio, lenguajes soportados) , etc., aunque no se indica las estrategias o técnicas aplicadas para su medición. Más de un indicador considerado en ambos frameworks se contempla una medición subjetiva de la eficiencia y eficacia como complemento para determinar la satisfacción y/o aplicabilidad de una técnica o herramienta de testing.

Los indicadores de uno de los frameworks estudiados (Cotroneo et al., 2013), difieren de los otros frameworks por lo menos a alto nivel; así: La habilidad para detección de defectos se determina a partir de: porcentaje de defectos detectados independientemente de su tipo; El costo de detección de defectos se determina por: defectos detectados en un tiempo dado respecto al esfuerzo (personas/hora); A su vez la actitud hacia el tipo de defectos es determinado por: porcentaje de defectos detectados por tipo y la propensión a detectar cierto tipo de defecto.

2.3.4. Instanciaciones

Respecto a los frameworks presentados en la Tabla 7, de acuerdo a la revisión de literatura, se han encontrado múltiples instanciaciones en la industria; tal es el caso de dos frameworks (Vos et al., 2011), (Vos et al., 2012) que han sido instanciados en la industria en proyectos reales, por lo que son reportados en distintos trabajos de investigación (Brosse et al., 2014), (Condori, Kruse, Vos, Brosse, y Bagnato, 2014), (Vos et al., 2011), (Vos et al.,

2012) relacionados con la evaluación y selección de técnicas de testing. Los demás frameworks han sido instanciados en ambientes académicos según se reporta en dos trabajos de investigación (Cotroneo et al., 2013), (Eldh et al., 2006).

2.3.5. Mantenimiento de la base de conocimientos

La base de conocimiento debe ser refinada con el tiempo mediante agregación iterativa de conocimiento producido en la organización (Cotroneo et al., 2013). Vegas y Basili (2005) proponen un mecanismo que sugiere una forma de mantener actualizada esta base de conocimiento o repositorio con la participación de un bibliotecario (encargado de registrar información) que de manera indirecta supervisa la información para el mantenimiento del repositorio; y de manera directa a partir de la información provista por:

- Los testers (consumidores) proveniente de sus experiencias en el uso de técnicas de testing.
- Investigadores en el área (productores) proveniente del resultado de sus investigaciones en el desarrollo de nuevas técnicas, así como en el estudio de las condiciones de aplicabilidad de las ya existentes.

Figura 2 – **Repositorio.**
Fuente (Vegas y Basili, 2005)

2.4. Estudios secundarios para la evaluación de técnicas de testing

Los estudios secundarios se realizan directamente en la labor de evaluación y selección de técnicas y/o herramientas de testing basándose en información existente proveniente de estudios primarios. Esto quiere decir que no es

necesario instanciar directamente una técnica o herramienta de testing o un conjunto de ellas para decidir sobre su uso para probar un sistema dado. Se trata de realizar comparaciones entre técnicas o herramientas a partir de información ya existente en una base de conocimiento producida por estudios primarios.

Es importante entender que para realizar estudios secundarios posiblemente sea necesario contar con suficiente y variada información sobre las técnicas de testing y su instanciación (estudios primarios).

Vos et al. afirman que, si los casos de estudio se ejecutan de acuerdo a un diseño común, será más fácil replicar los estudios y la evidencia creada será más fácil de comparar. Tal afirmación la realizan dado que su propuesta es un framework para casos de estudio, sin embargo, podría afirmarse que en general los estudios primarios deben ser diseñados bajo un mismo enfoque y artefactos (checklist, esquemas de caracterización, entre otros) para facilitar la comparación de la evidencia acerca de las técnicas de testing y su instanciación en estudios secundarios (Vos et al., 2012).

2.4.1. Problemas en el proceso de recomendación de técnicas y herramientas de testing de software

En los trabajos de recomendación de técnicas de testing, se encuentran varios problemas entre los cuales podríamos mencionar: la posible subjetividad de los testers para asignar importancia a los criterios de decisión, la presencia de datos con valores faltantes en los repositorios preexistentes a partir de los cuales se hará la sugerencia, múltiple terminología que podría usarse indicar los valores de los atributos para caracterizar una técnica o herramienta.

Para el método Porantim propuesto en una investigación (Dias y Travassos, 2009), no se pide a los testers los pesos de los atributos, ya estos se obtienen a partir de un estudio realizado con investigadores como parte de la construcción del esquema de caracterización. Thesaurus, es una herramienta desarrollada para normalizar la terminología utilizada en la caracterización de las herramientas de testing en cuanto a actividades, tareas y conceptos

utilizados en el proceso de desarrollo de software, cuyas equivalencias debe ser mantenida por un experto del dominio (Pilar, Simmonds, y Astudillo, 2014).

2.5. Sistemas de recomendación

De acuerdo con una trabajo de investigación elaborado mediante revisión de literatura (Bobadilla, Ortega, Hernando, y Gutiérrez, 2013), los sistemas de recomendación, apoyan la toma de decisiones en diversos dominios de aplicación, recomendando determinados ítems a los usuarios, en base a una recopilación de información acerca de sus preferencias respecto a tales ítems. La información puede ser adquirida de dos maneras:

- Explícitamente - generalmente en base a las calificaciones que los usuarios asignan a los ítems tras su experiencia de uso.
- Implícitamente - generalmente mediante monitoreo del comportamiento del usuario tal como: canciones que escucha, aplicaciones descargadas, sitios web visitados, libros leídos, entre otros. Los Sistemas de recomendación pueden usar características demográficas de los usuarios e información social, existiendo una fuerte tendencia hacia el usa de información a partir de internet de las cosas.

El proceso para generar recomendaciones se basa en una serie de aspectos como:

- El tipo de datos disponibles en su base de datos: calificaciones, información de registro de usuario, características y contenido sobre los ítems, relaciones sociales entre usuarios e información sobre su ubicación.
- El algoritmo de filtrado utilizado: demográfico, basado en contenido, colaborativo, basado en datos sociales, basado en contexto e híbridos
- El modelo elegido: basado en memoria el cual se basa directamente en datos o basado en un modelo generado a partir de los datos.

- Las técnicas empleadas: enfoques probabilísticos, redes bayesianas, algoritmo del vecino más cercano, redes neuronales, algoritmos genéticos, etc.
- Rendimiento del Sistema en cuanto al consume de tiempo y memoria.
- La calidad deseada de los resultados: novedad de los ítems, cobertura y precisión.

2.5.1. Filtrado colaborativo

El filtrado colaborativo está basado en el hecho que la opinión un grupo de usuarios es importante para la toma de decisión personal. A este grupo de usuarios se denomina los vecinos más cercanos, los cuales son usuarios que tienen preferencias o comportamientos similares respecto al usuario interesado en obtener una recomendación. Como base de datos, el filtrado colaborativo se basa en un conjunto de usuarios y un conjunto de ítems. La calificación que los usuarios asignan a un ítem es explotada en futuras sesiones de recomendación para predecir la calificación que un usuario pueda asignar a un ítem.

2.5.2. Basados en contenido

Los sistemas de recomendación basados en contenido tienen como base datos, un conjunto de usuarios y un conjunto de categorías o palabras claves que caracterizan a los ítems de manera que sea posible recomendar ítems basados en la caracterización de estos. Se calcula un conjunto de ítems parecidos a los ítems ya conocidos por el usuario actual, comparando el contenido de los ítems ya consumidos con los nuevos ítems que potencialmente le pueden ser recomendados.

2.5.3. Trust

La confianza y la reputación es un área importante de la investigación en Sistemas de recomendación; esta área está estrechamente relacionada con la información social actualmente incluida en los Sistemas de recomendación. Los enfoques más comunes para generar medidas de confianza y reputación

son las siguientes: (a) la confianza del usuario: para calcular la credibilidad de los usuarios a través de información explícita del resto de usuarios o para calcular la credibilidad de los usuarios a través de la información implícita obtenida en un red social y (b) la confianza en un ítem: para calcular la reputación de ítems a través de una retroalimentación de los usuarios o para calcular la reputación de ítems estudiando cómo los usuarios trabajan con estos ítems (Bobadilla et al., 2013).

2.5.4. Aplicación de sistemas de recomendación en ingeniería de software

Los desarrolladores de software utilizan diversas herramientas para realizar su trabajo, estas herramientas en un principio estuvieron orientadas a las funcionalidades de compilación y ensamblado, sin embargo en la actualidad se utilizan herramientas más sofisticadas para dar cobertura a todas las actividades del ciclo de vida, así mismo los sistemas de recomendación en el dominio de ingeniería de software, han surgido producto de la oportunidad de aprovechar los datos acumulados sobre experiencias en los distintos aspectos de la ingeniería de software con la finalidad de asistir a los desarrolladores en un contexto dado (Robillard, Maalej, Walker, y Zimmermann, 2014).

A diferencia de los Sistemas de recomendación tradicionales, que se basan en datos obtenidos a partir de las calificaciones que los usuarios realizan sobre los ítems, en ingeniería de software, el principal desafío radica en interpretar datos altamente técnicos almacenados en repositorios de software de manera automática.

Los sistemas de recomendación, hacen uso de distintas fuentes de información para ofrecer a los usuarios predicciones y recomendaciones de ítems. Estos tratan de balancear factores como exactitud, novedad, dispersidad, y estabilidad. Los métodos de filtrado colaborativo juegan un rol importante en la recomendación, aunque a menudo son usados con otras técnicas de filtrado como el basado en contenido, basado en conocimiento o sociales (Bobadilla et al., 2013).

Entre otros trabajos que utilizan filtrado colaborativo en el ámbito de la ingeniería de software, en encontró un método para realizar estimaciones de esfuerzo, para lo cual evalúa la similitud de un proyecto con otros proyectos anteriores para luego predecir el esfuerzo del proyecto objetivo (Ohsugi et al., 2004).

Se ha encontrado en la literatura, un sistema de recomendación que recomienda al desarrollador, archivos de clases de la librería Java que fueron utilizados en programas similares pero no en el programa actual del desarrollador (Pargament, Koenig, y Perez, 2000).

Así mismo en un estudio exploratorio (Robillard et al., 2014), se mencionan dos Sistemas de recomendación en el dominio de ingeniería de software: (1) Hipikat, apuntando a la evolución de software en proyectos de Código abierto, específicamente orientado a asistir a los desarrolladores novatos y (2) ImpRec, un sistema de recomendación que asiste el análisis del impacto del cambio critico en seguridad en empresas con procesos de software costosos.

2.5.5. Evaluación de sistemas de recomendación

En un artículo referente en cuanto a sistemas de recomendación en general (Bobadilla et al., 2013), se menciona que la evaluación de un sistema de recomendación contempla las siguientes medidas.

- Calidad de predicciones: error medio absoluto, exactitud, cobertura.
- Calidad del conjunto de recomendaciones: precisión, recall and F1(combinación de las dos primeras).
- Calidad de la lista de recomendaciones: medidas de ranking.
- Novedad y diversidad
- Estabilidad
- Confiabilidad

Existen otros trabajos orientados a la evaluación de sistemas de recomendación (Armentano, Christensen, y Schiaffino, 2015), (Avazpour, Pitakrat, Grunske, y Grundy, 2014), (Pu y Chen, 2011), (Said, Fields, Jain, y Albayrak, 2013) en los que se refiere y propone frameworks, categorías, dimensiones y métricas de evaluación. Estos trabajos toman como base los conocidos modelos TAM (Technology Aceptance Model), SUMI (Software Usability Measurement Inventory), los cuales contemplan la satisfacción del usuario, intención de uso, facilidad de uso percibida, entre otras dimensiones.

Se ha encontrado otros trabajos (Pilar et al., 2014), (Dias y Travassos, 2014), en los que proponen y validan métodos y herramientas para asistir en la selección de técnicas y/o herramientas de testing de software, en los cuales la validación ha sido realizada midiendo la exactitud de las sugerencias de técnicas o herramientas haciendo un contraste entre los resultados del método que proponen y un conjunto de verdad obtenido de las sugerencias de expertos.

2.6. Enfoques para la toma de decisiones

AHP: es una técnica para toma de decisiones que utiliza comparación por pares, así como el juicio de experto para derivar escalas de prioridad entre los criterios de selección. Las comparaciones son realizadas utilizando una escala de juicios absoluta que representa cuanto más un elemento domina a otro respecto a un atributo dado. Mediante la creación de una matriz de comparación de criterios, esta técnica genera un conjunto de pesos que representan el grado de preferencia entre criterios, luego se debe realizar una matriz de comparación por cada criterio. (Pilar et al., 2014).

TOPSIS: Es una técnica utilizada para ordenar las preferencias según su similitud con la solución ideal (Technique for Order preference by Similarity to Ideal Solution), y se usa para calcular las ventajas relativas a cada alternativa (individualmente); se considera que la mejor alternativa a elegir es aquella que tiene la distancia más corta hacia la solución ideal positiva y la más larga hacia la solución ideal negativa (Zaidan et al., 2015).

MAUT: (Multi Attribute Utility Theory), es un método utilizado como complemento a AHP de modo tal que luego de obtener los pesos que indican la escala de preferencia por los criterios de selección, se evalúa las alternativas utilizando una función de utilidad. MAUT asigna un valor de utilidad a cada acción que representa la preferencia hacia un criterio. De este modo se tendrá al final una matriz de preferencia en una única escala de cero a uno (Pilar et al., 2014).

2.7. Conclusiones del marco teórico

Pese a que existen varios esquemas de caracterización de técnicas y herramientas de testing de software, destaca un esquema de caracterización (Vegas y Basili, 2005), debido a su completitud y adaptabilidad a distintos contextos, lo cual es evidenciado por su instanciación en varios de los trabajos referidos a evaluación y selección de técnicas de testing de software; pese a ello entre los trabajos orientados a la evaluación de técnicas de testing, no existe suficiente uniformidad entre trabajos de investigación, ya que en general cada autor utiliza, adapta o propone un esquema de caracterización diferente, lo cual de por si impide aprovechar de manera conjunta los resultados de dichos estudios con miras a la realización de estudios secundarios.

Existen varios frameworks que orientan la conducción de estudios primarios, sin embargo uno de ellos (Vos et al., 2012) destaca en el diseño de casos de estudio y otro (Eldh et al., 2006) destaca en el diseño de experimentos; ambos presentan suficiente generalidad para poder ser adecuados en distintos contextos y a su vez suficiente detalle para la conducción de cada actividad.

En la revisión de literatura, la mayoría de frameworks para la selección o recomendación de técnicas de testing de software a manera de estudios secundarios, utilizan bases de conocimientos, repositorios o catálogos de técnicas de testing de software, sugiriendo en la mayoría de los casos que estos se van construyendo dentro de la organización a partir de la experiencia en la instanciación de las técnicas de testing.

No se ha encontrado trabajos de investigación que propongan la construcción y mantenimiento de repositorios de instanciaciones de técnicas de testing de software mediante la colaboración de múltiples organizaciones, lo cual permitiría contar con más información y conocimiento en menor tiempo y con menores costes de construcción y mantenimiento.

CAPÍTULO 3. METODOLOGÍA

3.1. Hipótesis de investigación

En base a los objetivos de la investigación, se plantean las siguientes hipótesis:

3.1.1. Hipótesis General

$\boldsymbol{H_i}$**:** "La calidad de las recomendaciones de técnicas de testing de software en base al método de recomendación de técnicas de testing SOTESTER, es buena"

3.1.2. Hipótesis Específicas

$\boldsymbol{H_{e1}}$**:** "La coincidencia entre expertos respecto a las técnicas de testing de software que recomiendan, será mayor a 1 en un rango de 0 a 3".

$\boldsymbol{H_{e2}}$**:** "La coincidencia del método propuesto SOTESTER con los expertos respecto a las técnicas de testing de software que recomiendan será mayor a 1 en un rango de 0 a 3".

3.2. Identificación y operacionalización de variables

Para la hipótesis $\boldsymbol{H_i}$, se ha identificado la Variable (X) "Calidad de las recomendaciones": referida a la calidad de recomendaciones de técnicas de testing de software realizadas por SOTESTER para un proyecto objetivo desde una perspectiva de la exactitud. La calidad recomendaciones viene dada por la medida en que SOTESTER recomienda ítems de manera similar

a como lo realiza un experto. Si los expertos coinciden entre sí, se espera que SOTESTER coincida con los expertos en la misma proporción.

- Definición operacional: La calidad de las recomendaciones, se medirá mediante análisis estadístico, comparando la mediana de las diferencias entre las variables “Experts” y “System”, donde: “Experts” es la coincidencia entre expertos respecto a las técnicas de testing de software que recomiendan, y “System” es la coincidencia entre el método propuesto SOTESTER con los expertos respecto a las técnicas de testing de software que recomiendan. La variable X, tomará solo dos valores: Buena, cuando la mediana de las diferencias entre “Experts” y “System” es igual a cero, y Mala, cuando la mediana de las diferencias entre “Experts” y “System” es igual diferente a cero.

Para la hipótesis $\boldsymbol{H_{e1}}$, se ha identificado la variable “Experts” que representa la coincidencia entre expertos respecto a las técnicas de testing de software que recomiendan.

- Definición Operacional: Se medirá comparando las 3 técnicas de testing de software que recomienda un experto con las 3 técnicas que recomienda cada uno de los otros expertos para un mismo proyecto objetivo. Por ejemplo, Si el experto 1 con el experto 2 no coinciden en ninguna técnica, entonces la variable toma el valor 0 para esa observación, y si ambos expertos recomiendan las mismas técnicas, entonces la variable toma el valor de 3 para esa observación.

Para la hipótesis $\boldsymbol{H_{e2}}$, se ha identificado la variable “System”, que representa la coincidencia entre el método propuesto SOTESTER con los expertos respecto a las técnicas de testing de software que recomiendan.

- Definición Operacional: Se medirá comparando las 3 técnicas de testing de software que recomienda el método SOTESTER con las 3 técnicas que recomiendan cada uno de los expertos para un mismo proyecto objetivo. Por ejemplo, si SOTESTER con un experto no coinciden en ninguna técnica, entonces la variable toma el valor 0 para esa observación, y si SOTESTER y

el experto recomiendan las mismas técnicas, entonces la variable toma el valor de 3 para esa observación.

3.3. Matriz de consistencia

ASPECTOS GENERALES			ASPECTOS ESPECIFICOS				
PROBLEMA GENERAL	**OBJETIVO GENERAL**	**HIPÓTESIS GENERAL**	**PROBLEMAS ESPECIFICOS**	**OBJETIVOS ESPECIFICOS**	**HIPOTESIS ESPECIFICAS**	**VARIABLES**	**Técnica de recolección de datos**
¿Cuál es la calidad de las recomendaciones de técnicas de testing de software realizadas mediante el Método propuesto SOTESTER respecto a las recomendaciones de técnicas de testing de software realizadas por expertos?	Medir la calidad de las recomendaciones de técnicas de testing de software realizadas por el método propuesto SOTESTER respecto a las recomendaciones de técnicas de testing de software realizadas por expertos.	La Calidad de las recomendaciones de técnicas de testing de software en base al método de recomendación de técnicas de testing SOTESTER, es buena.	¿En qué medida coinciden los expertos respecto a las técnicas de testing de software que recomiendan?	Medir la coincidencia de los expertos respecto a las técnicas de testing de software que recomiendan.	La coincidencia entre expertos respecto a las técnicas de testing de software que recomiendan, será mayor a 1 en un rango de 0 a 3.	Experts: Coincidencia de los expertos respecto a las técnicas de testing de software que recomiendan.	Cuestionario: Se utiliza un cuestionario para recopilar las recomendaciones de técnicas de testing de software que realizan los expertos para cada proyecto objetivo. Luego se procede a su comparación.
			¿En qué medida coincide el método propuesto SOTESTER con los expertos respecto a las técnicas de testing de software que recomiendan?	Medir la coincidencia del método propuesto SOTESTER con los expertos respecto a las técnicas de testing de software que recomiendan.	La coincidencia del método propuesto SOTESTER con los expertos respecto a las técnicas de testing de software que recomiendan será mayor a 1 en un rango de 0 a 3.	System: Coincidencia del método propuesto SOTESTER con los expertos respecto a las técnicas de testing de software que recomiendan.	Observación directa: Se observa las recomendaciones que realiza el método propuesto SOTESTER para cada proyecto objetivo, utilizando la aplicación web que lo implementa. Luego se procede a su comparación.

3.4. Tipo y Diseño de Investigación

El tipo de investigación es aplicada no experimental, dado que se emplean aspectos teóricos ya existentes referidos a sistemas de recomendación, y caracterización de técnicas de testing de software.

El alcance de la investigación es descriptivo, dado que no se busca establecer correlación o explicación entre variables. Se busca describir el resultado de aplicar un método propuesto. En el presente caso, se pretende determinar la calidad de las recomendaciones de técnicas de testing ofrecidas por el método propuesto SOTESTER, respecto a las recomendaciones que ofrecen expertos humanos.

El diseño es transversal - descriptivo, dado que se realizan observaciones simples a cada unidad de análisis (proyectos de software) en un único momento.

3.5. Unidad de análisis

La unidad de análisis está dada por los proyectos objetivo (proyecto de software para el cual se desea obtener recomendaciones de técnicas de testing de software).

3.6. Población de estudio

La población está dada por aquellos proyectos de software para los cuales los profesionales de testing de software se encuentran interesados en obtener recomendaciones de técnicas de testing de software. Estos proyectos son denominados proyectos objetivo.

Dada la intencionalidad del método propuesto (diversidad de ítems para los cuales se realizará la recomendación de técnicas de testing), la población de estudio es de características variables teniendo en cuenta los siguientes atributos: tipo de técnica de testing requerida, tipo de software a ser probado, fase de desarrollo o ciclo de vida en el que se realizará el testing, tamaño del

sistema a ser probado, entradas están disponibles para el proceso de testing, conocimiento del equipo de testing, salidas que el equipo de testing espera de la aplicación de técnicas de testing, tipos de defectos que se quiere detectar, aplicabilidad en tareas, forma de obtención de herramienta (opción comercial) y entorno (lenguaje de programación para el desarrollo del software). Los valores posibles de los atributos que definen la población de estudio están dados por el esquema de caracterización utilizado como parte del método propuesto, según se muestra en la *Tabla 8*.

Tabla 8 - **Esquema de caracterización adoptado en SOTESTER**

Atributo	Valores posibes
Estática o dinámica (Tipo de técnica de testing requerida))	Estática, dinámica
Tipo de software (tipo de software a ser probado)	Web, Web 2.0, código ISO C99, Mobile, Sistema de tiempo real, Batch, Sistema experto.
Fase del ciclo de vida (Fase de desarrollo o ciclo de vida en el que se realizará el testing)	Pruebas unitarias, pruebas de componentes, pruebas de integración de componentes, pruebas de integración del sistema, pruebas de aceptación, pruebas de regresión.
Escalabilidad (¿Qué tamaño tiene el Sistema a ser probado?, En líneas de código ejecutable.	<= 100, 101-300, 301-500, 501-1000, 1001-3000, >= 3001
Entradas (¿Qué entradas están disponibles para el proceso de testing?	Código Fuente, programa ejecutable, escenarios de ejecución, casos de prueba, registros de ejecución, requerimientos de la aplicación, limites inferiores y superiores para las variables.
Conocimiento (¿Qué conocimiento tiene e equipo de testing?)	Computacion evolutiva, definición y validación de funciones objetivo, dominio de la aplicación, , conocimiento acerca de cómo identificar el estado del Sistema a ser probado, identificación de escenarios, paradigma orientado a objetos, paradigma orientado a aspectos, codificación, generación de scripts de prueba, técnicas de categoría-partición, técnicas de gestión del proceso de pruebas, trabajo en equipo, liderazgo.
Experiencia (¿Qué Experiencia tiene el equipo de testing?)	Alguna experiencia con pruebas de cobertura, algoritmos evolutivos, pruebas de requerimientos, pruebas web.
Salidas (Salidas que el equipo de testing, espera de la aplicación de técnicas de testing)	Casos de prueba, información de cobertura, fallas.
Tipos de defectos (Principales tipos de defectos que se quiere detectar)	Assignment, Checking, Interface, Algorithm, Function, Timing/serialization, Build/package/merge, Documentación
Aplicabilidad en tareas (¿A qué tareas se quiere aplicar el testing?	Pruebas unitarias, pruebas de cobertura de código, pruebas de requerimientos, pruebas del sistema.
Obtención de herramienta (opción comercial)	Open source, Shareware of commercial, Comercial
Entorno – lenguaje de programación (lenguaje de programación para el desarrollo del software)	Java , Pascal, C, C++, C#, Cobol, Eiffel, Python, PHP, Perl, Simula, Visual Basic, Basic, Html, Ruby, Delphi/Object Pascal, Swift

Fuente. Elaborado en base a datos tomados tomados de (Dias y Travassos, 2014), (Vegas y Basili, 2005) y (Vos et al., 2012)

3.7. Tamaño de muestra

Se consideró un muestreo no probabilístico , tomando 11 proyectos objetivo, registrados en el repositorio de SOTESTER. Estos proyectos fueron

aportados por profesionales de la industria de software en el Perú y los detalles de su caracterización se encuentran en el Apéndice 3.

3.8. Técnicas de recolección de Datos

Se emplearán dos técnicas de recolección de datos: observación directa y cuestionario.

3.8.1. Observación directa

Se utiliza la técnica de observación directa para recopilar las recomendaciones realizadas por el método SOTESTER, observando las técnicas recomendadas para cada proyecto objetivo. Esta observación se apoya en un proceso automático soportado por la aplicación web, la cual escribe un log en un archivo de texto con las recomendaciones realizadas y los datos que sustentan cada recomendación para evidenciar el cumplimiento del método propuesto.

3.8.2. Cuestionario

Se utiliza un cuestionario para recopilar las recomendaciones de técnicas de testing realizadas por los expertos humanos para cada proyecto objetivo. Posteriormente estos resultados se procesan en una hoja de cálculo.

3.8.3. Procedimiento

Luego de realizar el modelo teórico del método de recomendación, el cual se presenta en la sección 4.3, se elaboró una aplicación web que lo implementa, de modo que sea factible su validación en un entorno real.

En las evaluaciones orientadas al punto de vista del usuario, el mecanismo de evaluación generalmente es la aplicación de cuestionarios a los usuarios que participan obteniendo recomendaciones del sistema de recomendación, sin embargo, a diferencia de sistemas que recomiendan películas u otros productos en los que el usuario podría dar un feedback casi inmediato acerca de los ítems recomendados, en el sistema de recomendación propuesto en el

presente trabajo, los ítems a recomendar son técnicas de testing y para poder obtener un feedback respecto a su aplicación en proyectos reales, se tendría que esperar un tiempo mayor además de contar con la predisposición de los usuarios a seguir participando en el estudio.

Se ha encontrado una forma de validar el método de recomendación propuesto, contrastando las recomendaciones realizadas por el sistema con las recomendaciones realizadas por tres expertos en testing de software; de manera análoga a la validación de un método para recomendar herramientas para el proceso de desarrollo de software realizado en un trabajo de investigación en Chile (Pilar et al., 2014).

El procedimiento para la recolección de datos consta de los siguientes pasos:

i. Inicializar el repositorio con técnicas de testing y proyectos históricos
ii. Registrar los proyectos objetivo en el repositorio y obtener recomendaciones de técnicas de testing.
iii. Solicitar la recomendación de técnicas de testing a un grupo de expertos para cada proyecto objetivo.
iv. Contrastar las recomendaciones del sistema con la de los expertos.

3.9. Análisis e interpretación de la información

Los datos se obtendrán de dos fuentes, la primera: las recomendaciones de los expertos, y la segunda: las recomendaciones de SOTESTER. Estos datos deben procesarse de la siguiente manera:

i. Tabular las técnicas recomendadas tanto por los expertos como por SOTESTER para cada proyecto objetivo
ii. Encontrar la diferencia entre expertos:
    a. Experto 1 vs. Experto2
    b. Experto 1 vs. Experto3
    c. Experto 2 vs. Experto3
    d. Promedio de coincidencia entre expertos
iii. Encontrar la diferencia de SOTESTER con los expertos

a. SOTESTER vs. Experto1
b. SOTESTER vs. Experto2
c. SOTESTER vs. Experto3

iv. Comparar la coincidencia entre expertos vs. Coincidencia de SOTESTER con los expertos

3.10. Instrumentos de recolección de datos

Respecto a los instrumentos de recolección de datos, se emplearon dos:

Cuestionario de recomendación – experto. Para obtener recomendaciones de los expertos humanos para cada proyecto objetivo. Tiene cuatro secciones: la primera muestra la descripción y caracterización del proyecto, la segunda solicita los pesos para los atributos de evaluación de las técnicas de testing, la cuarta muestra el listado de técnicas de testing y la última sección solicita la recomendación de tres técnicas, así como una justificación de cada recomendación. La ficha completa se encuentra en Apéndice 1.

Matriz de análisis de recomendaciones. Tomando como referencia y adecuando la matriz de ratings de Resnick (Resnick, Iacovou, Suchak, Bergstrom, y Riedl, 1994), se observa y analiza las recomendaciones hechas por SOTESTER, visualizando los proyectos históricos, la similitud de los proyectos históricos con el proyecto objetivo, la evaluación de las técnicas instanciadas, así como el número de instanciaciones y el ranking de técnicas para el proyecto objetivo. En la Figura 3 se muestra la estructura de la matriz de análisis de las recomendaciones realizadas por SOTESTER y en Apéndice 2 se presenta las matrices por cada proyecto objetivo.

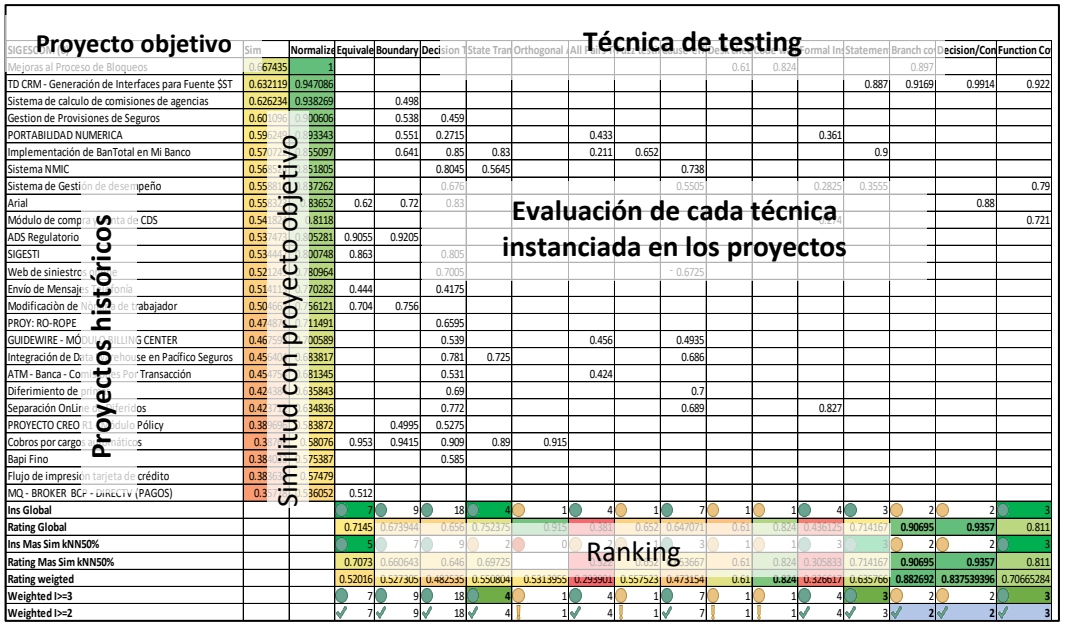

Figura 3 – **Estructura de matriz de análisis**
Fuente. Elaboración propia

3.11. Amenazas a la validez

Dado que el sistema ofrecerá recomendaciones a partir de un conjunto de caracterizaciones iniciales, es posible que se vea afectada la calidad de dichas recomendaciones si no se asegura la calidad de las caracterizaciones iniciales (del proyecto y técnicas instanciadas). Por ello se ha determinado un perfil mínimo el cual exige 3 años de experiencia mínima para los participantes que registran la data inicial del repositorio.

La validación del método de recomendación mediante el contraste de las recomendaciones hechas por el Sistema y las de expertos de la industria, podría no ser completamente objetiva en el sentido que podría haber variables no controladas tales como la experiencia, la diversidad de proyectos en que han trabajado, entre otros. Para minimizar esta amenaza, se considera un perfil de experto considerando como mínimo 10 años de experiencia en software testing y como mínimo experiencia en tres organizaciones diferentes.

En el futuro, puede considerarse una validación del método basado en sus propios resultados tales como nivel de confianza, precisión, intención de uso, satisfacción subjetiva, entre otros.

Es posible que el esquema de caracterización utilizado para caracterizar las técnicas de testing y los proyectos de software, no describan completamente el contexto del proyecto objetivo, sin embargo, la intención de la presente investigación es determinar la factibilidad básica del método y se espera hacer mejoras del esquema de caracterización en futuros trabajos de investigación

CAPÍTULO 4. DISEÑO DE SOTESTER

"Los Sistemas de Recomendación, asisten a los usuarios en el proceso de identificar ítems que cumplan sus deseos y necesidades. Estos sistemas son aplicados satisfactoriamente en diferentes configuraciones de e-commerce, por ejemplo en la recomendación de noticias, películas, música, libros y cámaras digitales" (Felfernig, Jeran, y Ninaus, 2013).

Son pocos los trabajos sobre sistemas de recomendación orientados al dominio de ingeniería de software y menos aun específicamente al dominio de testing de software, aunque si hay varios esfuerzos en general orientados a la selección de técnicas de testing de software mediante frameworks para la realización de estudios primarios y métodos para realizar estudios secundarios.

En el presente trabajo se propone un sistema de recomendación de técnicas de testing de software con un enfoque colaborativo que permita a profesionales de distintas organizaciones compartir su conocimiento y experiencia acerca del uso de técnicas de testing de software, dicha información deberá alimentar un repositorio a partir del cual el sistema hará recomendaciones de técnicas de testing de software según las características de un proyecto de software dado el desempeño de las técnicas en los proyectos existentes en el repositorio.

4.1. Trabajos relacionados

En la Tabla 9, se muestra un comparativo del presente trabajo con los trabajos relacionados más importantes encontrados en la literatura revisada.

Tabla 9 - **Comparativo con trabajos relacionados**

Aspectos	Trabajos relacionados	Este trabajo
Esquema de caracterización	(Vos et al., 2012): 22 atributos	21 atributos, se consiguió los valores posibles a partir de otros esquemas.
Repositorio/Catálogo	(Dias y Travassos, 2014; Vegas y Basili, 2005): En base al conocimiento de pocas personas (expertos) Catalogo estático: obtenido mediante revisión de literatura	En base al conocimiento de muchas personas: profesionales de testing sin o con experiencia (incluidos expertos) Catálogo dinámico (actualización continua). Memoria de recomendaciones otorga oportunidad de: investigación, afinamiento.
Método	(Dias y Travassos, 2014): Ranking de técnicas con análisis de combinación	Ranking de técnicas Sin análisis de combinación
Enfoques, Métodos, técnicas.	(Dias y Travassos, 2014): Similitud, kNN	Filtrado colaborativo Similitud, kNN TOPSIS, MAUT

Fuente. Elaboración propia.

4.2. Métodos de selección

Porantim, es un método y herramienta de selección de técnicas de testing específica para Model Based Testing (MBT) el cual, a partir de un repositorio de técnicas caracterizadas, permite establecer el nivel de adecuación de cada técnica al proyecto, así como el desempeño de cada técnica o combinación de técnicas (Dias y Travassos, 2014).

MENTOR es un método y herramienta para la selección de herramientas de testing usando métodos de toma de decisiones multicriterio AHP y MAUT (Pilar et al., 2014).

Vegas y Basili (2005) proponen un método para la selección de técnicas de testing de software mediante comparación usando un catálogo de técnicas elaborado en base a un esquema de caracterización. Los autores mencionan además el uso de una herramienta para su aplicación.

En el presente trabajo se presenta un método que, a diferencia de los trabajos mencionados, tiene un enfoque colaborativo para la alimentación del repositorio, y consta de un catálogo dinámico de proyectos e instanciaciones de técnicas. El método propuesto se aplica con el uso de una herramienta de sofware.

4.2.1. Esquema de caracterización

Se tomó como base principal, el esquema presentado por Vos et al. (2012), dado que ha sido instanciado en múltiples casos de estudio en la industria y se orienta a todo tipo de técnicas de testing, lo cual se corresponde con la finalidad del sistema de recomendación presentado en el presente trabajo. Este esquema de caracterización cuenta con 22 atributos, de los cuales ocho son considerados como “prerrequisitos”, cinco “de operación”, ocho “de resultados” y uno “de obtención de herramienta".

De manera complementaria y para determinar los valores posibles de cada atributo del esquema de caracterización se ha considerado dos de los esquemas de caracterización encontrados en la revisión de literatura (Vegas y Basili, 2005), (Dias y Travassos, 2014). En el primer esquema de caracterización (Vegas y Basili, 2005), los 32 atributos con que cuenta el esquema presentado, se hace referencia a dos grupos de atributos: el primero denominado “bounded variables”, comprende 6 atributos cuyos valores son impuestos por el proyecto y que no pueden cambiar durante la selección de técnicas; mientras que el segundo denominado “Non bounded Variables” comprende 26 atributos, cuyos valores pueden cambiar según las necesidades o preferencias en el proceso de selección. En el segundo (Dias y Travassos, 2014), del total de los 25 atributos presentados, se destaca un grupo de once atributos del esquema como “atributos de caracterización del proyecto”, los cuales sirven para determinar la similitud y por ende la adecuación de una técnica respecto a un proyecto dado, entendiéndose que los atributos restantes se refieren a la técnica de testing en sí.

En el presente trabajo, se tiene un total de 21 atributos de los cuales:

- 12 se agrupan como “atributos de similitud”, los cuales sirven para comparar proyectos entre sí. Por ejemplo, para comparar un proyecto de interés para el que se quiere obtener una recomendación con los proyectos existentes en el repositorio.
- Nueve se agrupan como “atributos de desempeño”, los cuales sirven para caracterizar las técnicas instanciadas en el proceso de testing de

un proyecto existente en el repositorio; o para indicar las preferencias de un usuario respecto al desempeño esperado de las técnicas que le serán recomendadas.

4.2.2. Repositorio

Son varias las investigaciones (Cotroneo et al., 2013), (Dias y Travassos, 2014), (Vegas, 2002), (Vegas y Basili, 2005), (Vos et al., 2011), (Vos et al., 2012) que mencionan la necesidad, o inclusive realizan esfuerzos para la creación de un repositorio o base de conocimiento para la evaluación y selección de técnicas y/o herramientas de testing de software.

Vegas y Basili (2005), tal como se muestra en la Figura 2, proponen un mecanismo que sugiere una forma de mantener actualizada la base de conocimiento, por un lado a partir de la información provista por los testers (consumidores) proveniente de las experiencias en el uso de técnicas de testing, y por otro lado a partir de información provista por investigadores en el área (productores) proveniente del resultado de sus investigaciones en el desarrollo de nuevas técnicas así como en el estudio de las condiciones de aplicabilidad de las ya existentes. Así mismo indican la necesidad de un bibliotecario que participará de manera indirecta supervisando la información para el mantenimiento del repositorio.

Se han encontrado dos repositorios disponibles: el de Vegas (2002) que a manera de catálogo logra caracterizar un total de 13 técnicas y por otro lado el de Días y Travassos (2009) que logra caracterizar un total de 219 técnicas o enfoques MBT. En ambos casos la caracterización de las técnicas o enfoques fue realizada mediante revisión de literatura técnica cuya evidencia varía entre: especulación, ejemplos, pruebas de concepto, reportes de experiencia industrial y experimentación.

La propuesta de repositorio del presente trabajo está basado en el propuesto por Vegas y Basili (2005), con las diferencias mencionadas en la

Tabla *10*.

Tabla 10 - **Diferencias entre repositorios**

Criterio de comparación	Repositorio de Vegas y Basili (2005)	Este trabajo
Actualización de catálogo	Estática: cada vez que se realiza una Revisión de literatura o se produce un nuevo trabajo de investigación sobre una o más técnicas.	Dinámica: cada vez que ingresa un nuevo proyecto al repositorio, las técnicas instanciadas son caracterizadas.
Actores que participan en el proceso	Consumidor: académico o de la industria Productor: académico que realiza un estudio primario (caso de estudio o experimento). Bibliotecario: académico que organiza, valida la información de los productores para alimentar, mantener y actualizar el repositorio	Consumidor: académico o de la industria. Productor: académico o de la industria que utiliza una técnica de testing.. (el mismo consumidor) Bibliotecario: no es necesario.
Cantidad de instanciaciones que alimentan el repositorio	Limitada a la cantidad de estudios primarios.	Cada vez que se instancia una técnica en la industria existe una oportunidad de alimentar el repositorio.

Fuente. Elaboración propia.

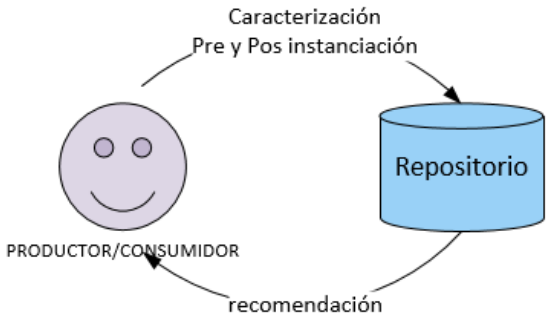

Figura 4 - **Repositorio propuesto en el presente trabajo**
Fuente. Elaboración propia

4.3. Descripción del método de recomendación

Se trata de un método colaborativo que según se muestra en la Figura 5, a partir de información proporcionada por la comunidad de testers acerca de la aplicación o instanciación de una o más técnicas de testing en un proyecto de software dado, se construye un repositorio de manera colaborativa. A partir de este repositorio, es posible previa caracterización de un proyecto, obtener un ranking de técnicas de testing aplicadas en proyectos similares, tal ranking es ordenado según el desempeño de las técnicas en los proyectos del repositorio, de modo que el tester o el equipo de testing pueda elegir la técnica

o técnicas a utilizar en su proyecto; posteriormente el tester o equipo de testing deberá caracterizar la instanciación de la(s) técnica(s) de testing empleada(s) en su proyecto para alimentar el repositorio y finalmente se determinará el nivel de certeza de cada recomendación, lo cual establecerá la reputación de cada una de las instanciaciones de técnicas participantes en las recomendaciones. El método consta de 4 pasos, de las cuales 2 son automatizados y 2 son manuales.

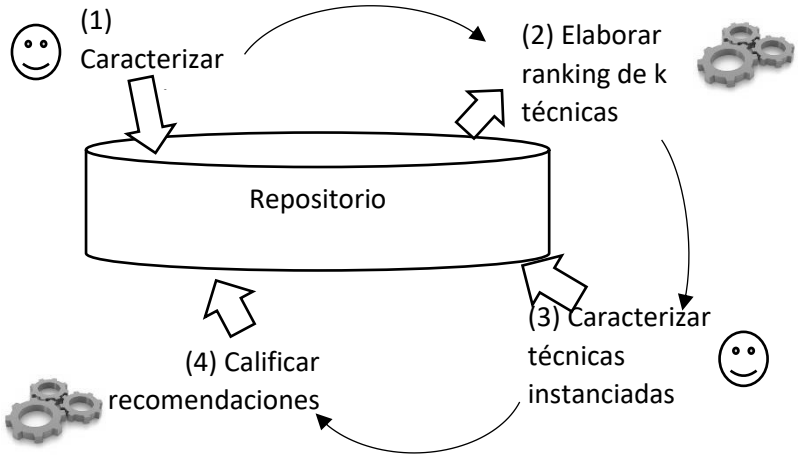

Figura 5 – **Pasos del método SOTESTER**
Fuente. Elaboración propia

4.3.1. Paso 1 - Caracterización del proyecto

El usuario deberá registrar las características del proyecto (por ejemplo: plataforma utilizada para el desarrollo, entradas disponibles para el testing, tipo de sistema a probar, entre otros.) para ello se empleará los atributos de similitud del esquema de caracterización según se mostró en la Tabla 8.

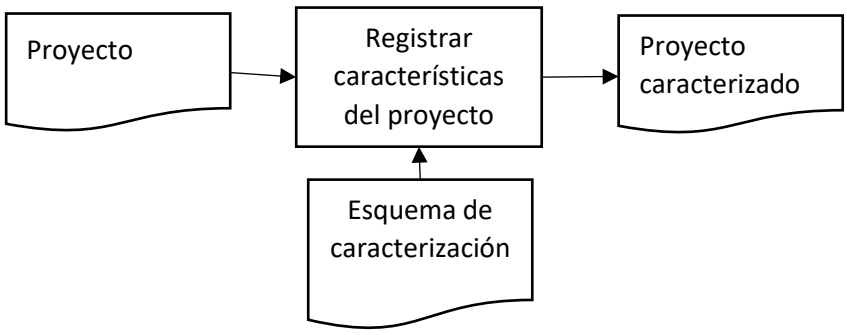

Figura 6 – **Caracterización del proyecto**
Fuente. Elaboración propia

4.3.2. Paso 2 - Elaborar ranking de técnicas

Para elaborar el ranking de técnicas a recomendar, se utiliza el algoritmo de filtrado colaborativo k Nearest Neighbor (kNN), según se indica en la Figura 7 y se detalla en las líneas siguientes.

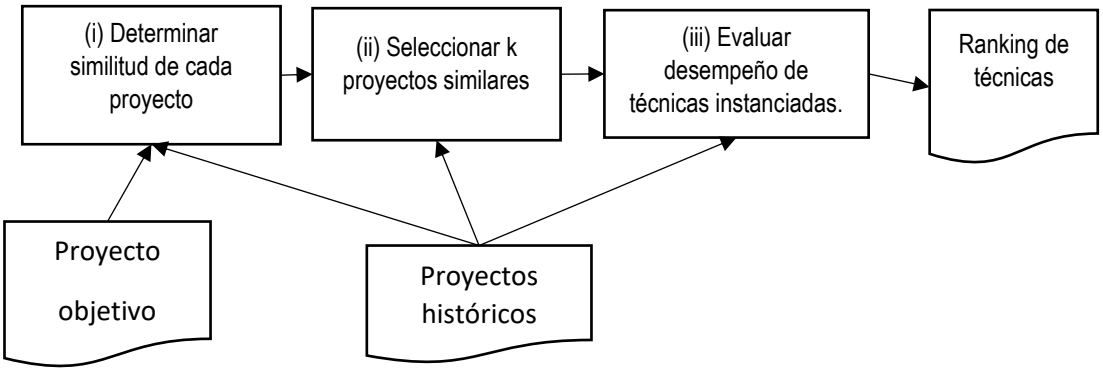

Figura 7 – **Elaboración de ranking de técnicas**
Fuente. Elaboración propia

En la primera fase del paso 2, se determina la similitud de cada proyecto en el repositorio con el proyecto de interés utilizando para ello el método TOPSIS, el cual ubica la puntuación de cada alternativa basado en su distancia euclidiana a las soluciones ideales positiva y negativa de manera análoga a lo realizado en uno de los antecedentes (Zaidan et al., 2015) al presente trabajo. Según esta técnica, la mejor alternativa es aquella con distancia más corta a la solución ideal positiva y con la más extensa a la solución ideal negativa. Para calcular la similitud, se sigue el siguiente procedimiento:

i. Encontrar Similitud relativa. En primer lugar se debe representar de manera vectorial, tanto el proyecto objetivo como cada uno de los proyectos del repositorio de manera análoga a lo que realizan Días y Travassos (2014) en el método Porantim, según se muestra en la Figura 8. Para ello se convierte los valores de cada atributo a valores numéricos utilizando el coeficiente de similitud “Coeficiente de Jaccard” según el algoritmo indicado en la Figura 9., multiplicándolo por el peso del atributo, basado en MAUT - Multiple Atribute Utility Theory (Teoria de Utilidad Multi Atributo) de manera análoga a otro estudio previo (Pilar et al., 2014).

$V = (X_1|X_2|X_3|\ldots|X_n)$
$V: es\ el\ vector\ que\ representa\ un\ proyecto\ de\ software$
$Xi: representa\ el\ valor\ del\ proyecto\ de\ software\ relacionado\ al\ atributo\ i$
$n: numero\ de\ atributos\ de\ similitud\ del\ esquema\ de\ caracterización$

Figura 8 - **Representación vectorial de un proyecto**
Fuente. Elaboración propia

APO: Atributo del Proyecto objetivo
APR: Atributo del Proyecto del repositorio
SR: Similitud relativa a un atributo
#: Número de elementos
W: Peso del atributo, el cual se obtiene del promedio simple de los pesos asignados al atributo por los usuarios del repositorio.

Si (APR=APO)
    Entonces SR ←1
De lo contrario, Si (APR ⊂ APO)
    Entonces SR ← (#APR/#APO)*W
De lo contrario
    SR← 0

Figura 9 - **Cálculo de similitud relativa**
Fuente. Elaboración propia

Es importante definir variables y expresiones a utilizar:

V^+: Vector que representa la solución ideal positiva, en este caso está conformado por los atributos de similitud del proyecto objetivo para el cual se realizará la recomendación. Los valores numéricos de cada atributo se corresponden directamente con los pesos W (promedio de los pesos asignados por los usuarios del repositorio), de manera que, si calculamos la utilidad del Vector en un rango de 0 a 1, esta sería igual a 1, lo cual indicaría que se trata de un proyecto idéntico al proyecto objetivo.

V^-: Vector que representa la solución ideal negativa, su utilidad sería igual a 0 y representa la situación en que no existen coincidencias con el proyecto objetivo.

V_j: Vector que representa a cada proyecto del repositorio, donde j toma valores desde 1 hasta m, siendo m el número total de proyectos en el repositorio.

$d(V_j, V^+)$: Distancia de la solución ideal positiva

$d(V_j, V^-)$: Distancia de la solución ideal negativa

ii. Encontrar la distancia de cada proyecto a la Solución ideal positiva

$$d(V_j, V^+) = \sqrt{\sum_{i=1}^{n} (X_i - X^+{}_i)^2}$$

iii. Encontrar la distancia de cada proyecto a la Solución ideal negativa

$$d(V_j, V^-) = \sqrt{\sum_{j=1}^{n} (v_j - v^-{}_j)^2}$$

iv. Encontrar la cercanía a la solución ideal.

$$C^+{}_j = \frac{d(V_j, V^-)}{d(V_j, V^-) + d(V_j, V^+)},\ 0 < C^+{}_j < 1$$

En la segunda fase del paso 2, se seleccionan los k proyectos históricos más parecidos al proyecto objetivo, para ello en lugar de contar con un valor concreto, en el método propuesto, se considera un porcentaje (50%), por lo que el conjunto de vecinos más cercanos estará conformado por aquellos proyectos similares al proyecto objetivo por lo menos en un 50%.

En la tercera fase del paso 2, se evalúa el desempeño de las técnicas instanciadas en los proyectos seleccionados.

i. El usuario indica pesos de preferencia para cada atributo de desempeño: W_l., donde $l = 1..p$, y p es la cantidad de atributos de desempeño del esquema de caracterización.
ii. Se excluyen las técnicas que no han sido instanciadas por lo menos *m* veces, donde *m* es la cantidad promedio de instanciaciones de las técnicas registradas en el repositorio.
iii. Basados en MAUT, para cada instanciación h extraída se calcula su utilidad en base a los atributos de desempeño del esquema de caracterización y a las preferencias del proyecto objetivo

$$U_h = \sum_{l=1}^{p} X_{hl} W_l \ .$$

iv. Se calcula la utilidad media por cada técnica a partir de la utilidad y confianza de cada instanciación *h*.

$$\overline{U} = \sum_{h=1}^{t} U_h \, \bar{C}_h$$

Donde:

- t: total de instanciaciones seleccionadas de una técnica dada;
- $\bar{C}_h$: nivel de confianza medio de las recomendaciones dadas por *h.*

Quedando así elaborado un ranking a partir del cual se recomienda al usuario las primeras Z técnicas con mayor utilidad media, donde Z es un valor ingresado por el usuario. Es importante tener claro que, en un principio al no contar con recomendaciones previas, todas las instanciaciones de técnicas en proyectos históricos cuentan con el mismo nivel de confianza igual a 1.

4.3.3. Paso 3 - Caracterizar técnicas instanciadas

Como parte del proceso de testing en un proyecto de software se instancian una o más técnicas recomendadas, por lo que, como se puede ver en la Figura 10 en este paso, el usuario a modo de feedback debe caracterizarlas en base a los atributos de desempeño. Las caracterizaciones de las técnicas instanciadas formarán parte del repositorio y podrán participar en el proceso de recomendación. El esquema de caracterización utilizado se muestra en la *Tabla 11*.

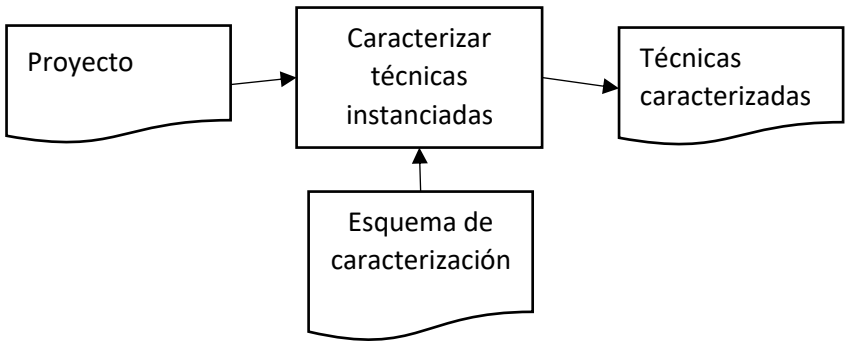

Figura 10 – **Caracterización de técnicas instanciadas**
Fuente. Elaboración propia

Tabla 11 - **Esquema de caracterización - Atributos de desempeño**

N°	Atributo	Descripción/ texto ayuda
1	Completitud	Cobertura provista por los casos de prueba
2	Eficacia	Capacidad de encontrar defectos
3	Tamaño del banco de pruebas	Nivel de adecuación del número de casos de prueba generados por unidad de tamaño de software
4	Interacción	Cuán adecuados son los modos de interacción; (Por ejemplo: navegación, consultas, refinamiento sucesivo, etc.) soportados por la técnica
5	Guía o participación del usuario	Cuan apropiada es la guía o participación que la técnica requiere del usuario. Por ejemplo: Nada apropiada (la técnica es completamente manual), evaluación manual de las salidas, selección de entradas apropiadas, definición de patrones, filtrado, etc.
6	Fuentes de información	Disponibilidad de fuentes de información acerca de la técnica
7	Comprensibilidad	Si la técnica es o no es fácil de entender
8	Satisfacción subjetiva	Respecto a la técnica
9	Esfuerzo	¿Cuánto esfuerzo toma aplicarla (esfuerzo en aprendizaje, instalación, configuración y ejecución)?

Fuente. Datos tomados de (Vos et al., 2012).

4.3.4. Paso 4 - Calificar recomendaciones

Consiste en determinar el error de las recomendaciones para una técnica instanciada en el proyecto objetivo. Este error servirá para determinar el nivel de confianza al ofrecer otras recomendaciones.

$$E_r = Abs(U_h - U_o)$$

U_h: Desempeño de la técnica según la instanciación *h* utilizada en la recomendación *r*

U_o: Desempeño de la técnica en el proyecto objetivo *o*

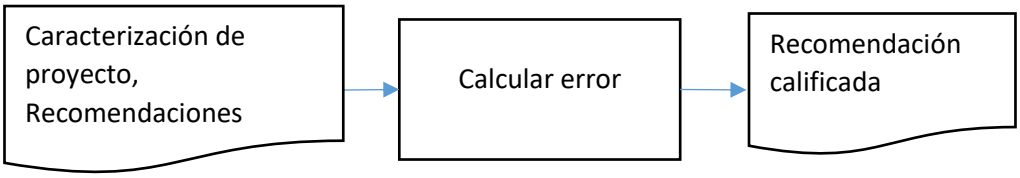

Figura 11 – **Calificación de recomendaciones**
Fuente. Elaboración propia

4.4. Construcción de la herramienta SOTESTER

Para facilitar la implementación del método SOTESTER, se construyó una aplicación web que implementa cada uno de los pasos del método, permitiendo que los profesionales de testing de software puedan crear una

cuenta y accedan para aportar información acerca de su experiencia en la instanciación de técnicas de testing en proyectos históricos, así como para obtener recomendaciones de técnicas de testing para proyectos objetivos actuales.

4.4.1. Requisitos

Requisito de negocio. Contribuir a asegurar la eficiencia y eficacia del proceso de testing de software en un proyecto dado mediante una adecuada selección de técnicas de testing de software.

Requisitos funcionales.

- RF1 El sistema debe permitir el auto registro de nuevos usuarios, para los cuales creará una cuenta de usuario asociada a una organización.
- RF2 El sistema debe permitir registrar los pesos de preferencia del usuario respecto a los atributos de similitud del esquema de caracterización vigente.
- RF3 El sistema debe permitir registrar proyectos en dos modalidades: (1) en modo catalogador y (2) en modo normal.
- RF4 El sistema debe permitir caracterizar un proyecto teniendo en cuenta un único esquema de caracterización configurado como “Vigente”. Los atributos de caracterización pueden ser de opción múltiple o de opción simple.
- RF5 El sistema debe permitir al usuario obtener recomendaciones mediante ranking de las k técnicas más adecuadas para un proyecto.
- RF6 El sistema debe permitir al usuario registrar caracterización de cada técnica usada o instanciada en un proyecto. Los atributos de caracterización pueden ser valores numéricos, de opción simple, de opción múltiple, de texto abierto.
- RF7 El Sistema debe permitir al usuario cambiar el estado de sus proyectos según sea el estado actual en que se encuentren.
- RF8 El sistema debe permitir al usuario solicitar la recomendación de técnicas de testing en número variable, para lo cual deberá solicitar además de la longitud del ranking, los pesos de sus preferencias

respecto a los atributos de desempeño de la técnica teniendo en cuenta el esquema de caracterización vigente.

- RF9 El sistema debe calificar automáticamente las distintas recomendaciones realizadas una vez que las técnicas instanciadas en un proyecto de interés hayan sido caracterizadas.
- RF10 El sistema debe mostrar al usuario una tabla con los proyectos de su propiedad con la siguiente información: nombre, descripción, fecha de inicio, fecha estimada de culminación, estado actual, acciones posibles.
- RF11 El sistema debe permitir al usuario ver de manera detallada un proyecto seleccionado mostrando un historial de sus estados y la información referida a cada uno de sus estados.

Requisitos no funcionales.

- RNF1 Disponibilidad : 24/7 a un 90%
- RNF2Seguridad:
  - Se debe manejar autenticación de cada usuario.
  - Se debe contar una base de datos de seguridad que permita gestionar usuarios y los roles que determinen los accesos de los usuarios a las funcionalidades de la aplicación. La gestión de roles debe ser escalable, aunque en principio como mínimo debe contemplar los siguientes:
  - Tester: Acceso a todas las funcionalidades excepto el mantenimiento de usuarios y roles
  - Administrador: funcionalidad de mantenimiento de usuarios y roles
- RNF3 Mantenibilidad:
  - Documentar código utilizando XML.
  - Desarrollo en capas
  - Los distintos artefactos deben ser construidos en el idioma inglés.
- RNF4 Confidencialidad
  - No debe ser posible que un usuario que no sea el dueño de un proyecto dado acceda a información de otra organización y a los proyectos de dicha organización. La autoría de la información de los proyectos

utilizada para realizar recomendaciones no debe ser identificable por un usuario diferente al propietario.

- RNF5 Usabilidad:
  - El idioma de la GUI de la aplicación debe ser inglés y español
  - La aplicación web debe ser responsiva.
  - La aplicación debe contar con un manual de usuario disponible en línea.
  - La aplicación debe mostrar descripciones para cada una de las acciones a seguir por el usuario, además las etiquetas de cada entrada de datos en la GUI.

Restricciones.

- N° de usuarios en horas de más alto tráfico: 10 (Contemplar como funcionalidad critica: obtener recomendación)
- Tecnología:
  - IDE: Se debe utilizar Visual Studio 2015 Community
  - Manejador de Base de datos: Sql server 2014 Estándar Edition.
  - Lenguaje de programación: c#
  - Frameworks: ASP.NET, Entity Framework 6.0
  - Patrones y estilos: Debe emplearse el patrón MVC

4.4.2. Casos de uso

Se ha identificado seis casos de uso y tres actores. En la Tabla 12 se describen los actores de los casos de uso del sistema y en la Figura 12 se muestra la relación entre actores y los casos de uso.

Tabla 12 - **Actores de casos de uso**

Nombre	Descripción	Casos de uso asociados
Practitioner	Representa a una organización que tiene a su cargo proyectos de software y sus respectivas actividades de testing.	
Project owner	representa a la organización propietaria del proyecto	Caracterizar proyecto, Obtener recomendación, caracterizar técnica ya catalogada, calificar recomendación.

Pioneer	Son usuarios que caracterizarán las técnicas de software por primera vez en el repositorio.	Caracterizar técnica no catalogada

Fuente. Elaboración propia.

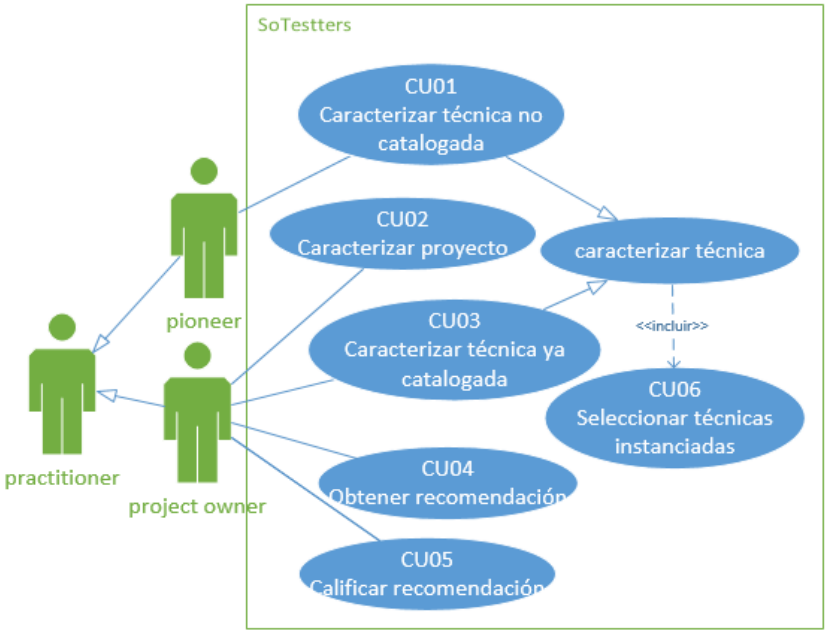

Figura 12 – **Casos de uso**.
Fuente. Elaboración propia

4.4.3. Clases de negocio

Tal como se muestra en la Figura 13, las clases contempladas permitirán gestionar los datos y acciones referidos a los proyectos de software, su caracterización y la de las técnicas instanciadas, así como la evaluación de las recomendaciones realizadas por el sistema. En la Tabla 13 se describe las clases de negocio.

Tabla 13 - **Clases de negocio**

Clase	Descripción
Technique	Técnica de testing de software.
Characterization Schema	Esquema de caracterización de técnicas de testing, consta de un conjunto de atributos.
Attribute	Atributo o característica perteneciente a un esquema de caracterización, puede ser de similitud o de desempeño y puede o no tener un conjunto de valores posibles asociados.
AllowedValue	Valores permitidos para cada atributo cuando corresponda.
AdequacyWeight	Representa a importancia o peso indicado por el usuario sobre un atributo de similitud, el valor del peso o la importancia del atributo, se utiliza para determinar la similitud entre proyectos.
Project	Un proyecto de software perteneciente a un usuario, para el cual se pueden obtener recomendaciones en base a su similitud con otros proyectos existentes en el repositorio.

ProjectState	Representa el estado de un proyecto, puede ser: no caracterizado, en caracterización antes de instanciación, obteniendo recomendación, caracterizando después de instanciación, cerrado.
Characterization	Caracterización de un proyecto de software y de cada técnica instanciada en el proceso de testing.
CharacterizationDetail	Valor por cada atributo de una caracterización, puede ser un valor seleccionado entre un conjunto de valores posibles, o un valor numérico ingresado por el usuario.
Organization	Usuario (entiéndase que cada organización participante cuenta con una cuenta de usuario).
Recommendation	Recomendación mediante ranking de técnicas de testing de software realizada por el sistema para un proyecto dado. Cada instanciación de técnica contemplada para elaborar el ranking, se considera una recomendación.
PreferenceWeight	Representa la preferencia de los usuarios respecto a los atributos de desempeño de una técnica de testing. Estas preferencias se utilizan en el proceso de evaluación de las técnicas instanciadas en los proyectos similares a un proyecto dado para el cual se obtiene recomendaciones.

Fuente. Elaboración propia.

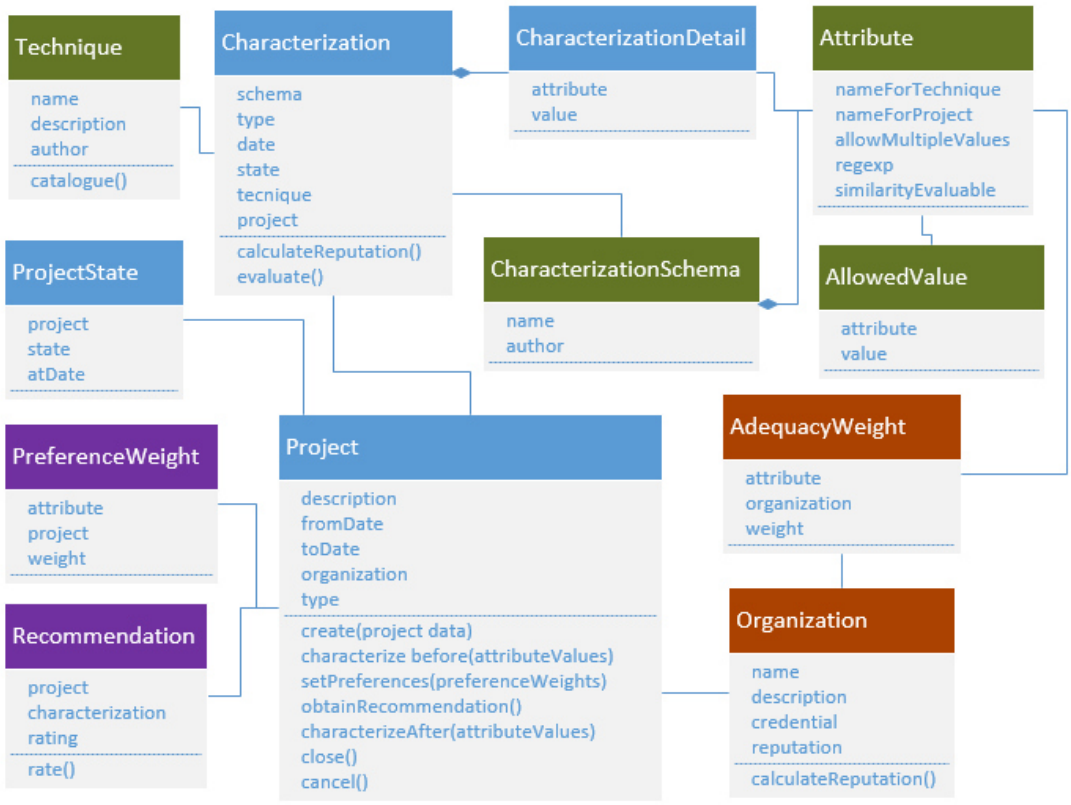

Figura 13 – Diagrama de clases de negocio
Fuente. Elaboración propia

4.4.4. Estados de un proyecto

Se ha identificado ocho estados posibles para un proyecto de software en el repositorio. Estos estados permiten al sistema, controlar las acciones que se puede realizar para cada proyecto; en la Tabla 14, se describe cada uno de estos estados y para facilitar su comprensión, en la Figura 14, se muestra un diagrama que representa el flujo de la transición de estados.

Tabla 14 - **Estados de un proyecto**

N°	Estado	Descripción
1	Non characterized (No caracterizado)	Es el estado inicial de un proyecto recién creado en el repositorio. En dicho estado el proyecto solo cuenta con datos generales, tal como un nombre, una descripción, fecha de inicio y fecha estimada de fin.
2	Characterizing before instantiation (en caracterización)	Es el estado en el cual se ingresan los valores para los atributos de caracterización del proyecto, de modo que sea posible para compararlo con otros proyectos del repositorio y establecer su similitud.
3	In testing (en testing)	Es el estado en el cual se encuentra un proyecto luego de obtener una recomendación. Un proyecto que se ha ingresado al repositorio para catalogar por primera vez alguna técnica o para la inicialización del repositorio, nunca se encontrará en este estado.
4	Selecting instanced Technique (seleccionando las técnicas instanciadas)	Es el estado en el cual el usuario debe indicar al sistema cuales de las técnicas recomendadas, fueron efectivamente instanciadas durante el proceso de testing en su proyecto de software.
5	Characterizing after instantiation (caracterizando técnicas pos instanciación)	Es el estado en el que se encuentra el proyecto una vez indicadas las técnicas instanciadas. En este estado el usuario debe caracterizar o evaluar cada técnica instanciada en su proyecto empleando para ello los atributos de desempeño del esquema de caracterización.
6	Evaluating recomendation (Evaluando recomendación)	Es el estado en el que el usuario debe evaluar cada recomendación obtenida del sistema, además internamente el sistema determinará el grado de certeza de la recomendación en base a la información obtenida en el estado anterior. Los proyectos catalogadores de una técnica por primera vez o aquellos que sirven para inicializar el repositorio, nunca se encontrarán en este estado.
7	Closed (Cerrado)	Es el estado final al que llegan todos los proyectos válidos. En este estado, los proyectos son tomados en cuenta para obtener recomendaciones.
8	Canceled (Cancelado)	Es el estado final al que llegan los proyectos no validos por decisión propia del usuario o por abandono (cuando en un plazo determinado, no ha continuado hacia el estado siguiente).

Fuente. Elaboración propia.

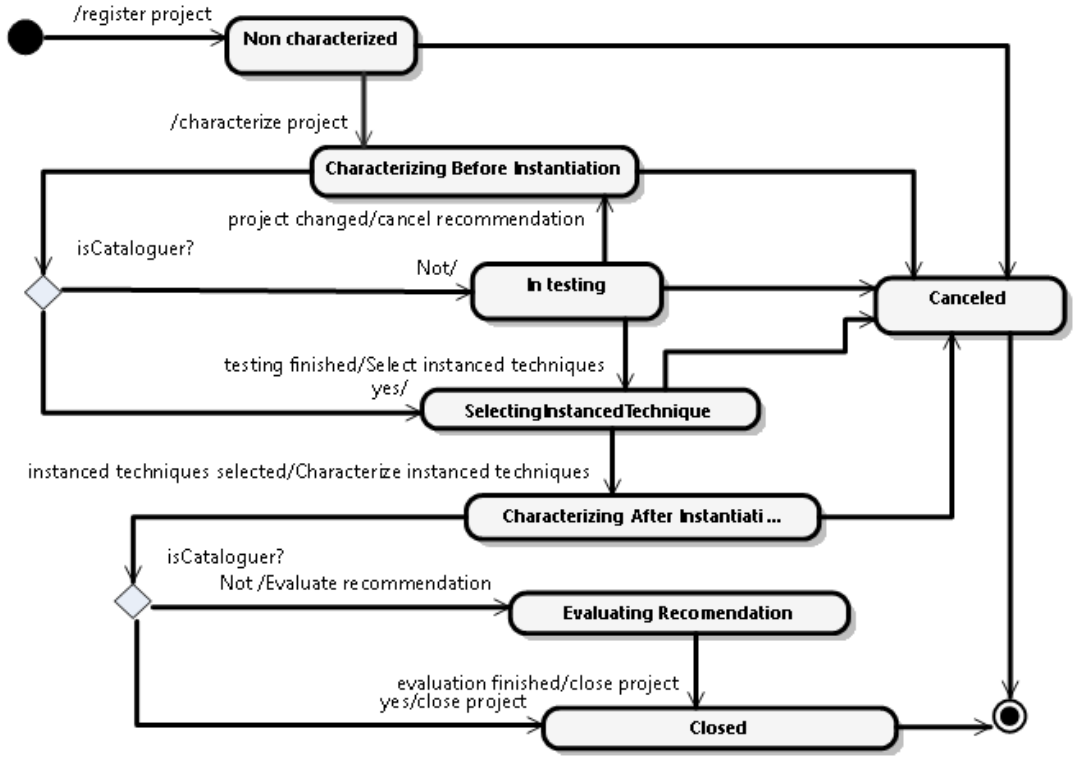

Figura 14 – **Diagrama de transición de estados de un proyecto.**
Fuente. Elaboración propia

4.4.5. Vista de desarrollo

En esta vista se muestran aspectos relacionados a la construcción del software, tal como la organización de archivos de código fuente, el contenido de estos archivos y la relación entre ellos. La Tabla 15 describe de manera general los archivos de código fuente y la Figura 15 su organización, en ambos casos se toma como ejemplo solo la entidad “Project”.

Tabla 15 - **Principales archivos de código fuente**

Archivo	Descripción
soTesterEF.edmx	Es el archivo con el modelo de persistencia de datos del framework de persistencia “Entity framework”, a partir del cual se obtiene las clases que se corresponden con cada una de las tablas en la base de datos.
*.cshtml	Los archivos con esta extensión, corresponden a las vistas que para su programación utilizan html y c#. En general todas las vistas utilizan Razor.
*Controller.cs	Archivos de código fuente en lenguaje c#. Los controladores sirven para enlazar las vistas con los modelos.
Web.config	Archivo de configuración que contiene información como; cadena de conexión, proveedores de datos, roles, entre otros.
Project.cs	Existen dos archivos del mismo nombre El primero, corresponde al código autogenerado por entityframework mediante el mapeo de la tabla de base de datos correspondiente a la entidad “Project”.

	El segundo, es código programado manualmente en el cual se implementan los distintos métodos de negocio de la entidad “Project”. De manera análoga, se cuenta con dos archivos de código fuente por cada entidad (Technique, Characterization, Recommendation, entre otras).

Fuente. Elaboración propia.

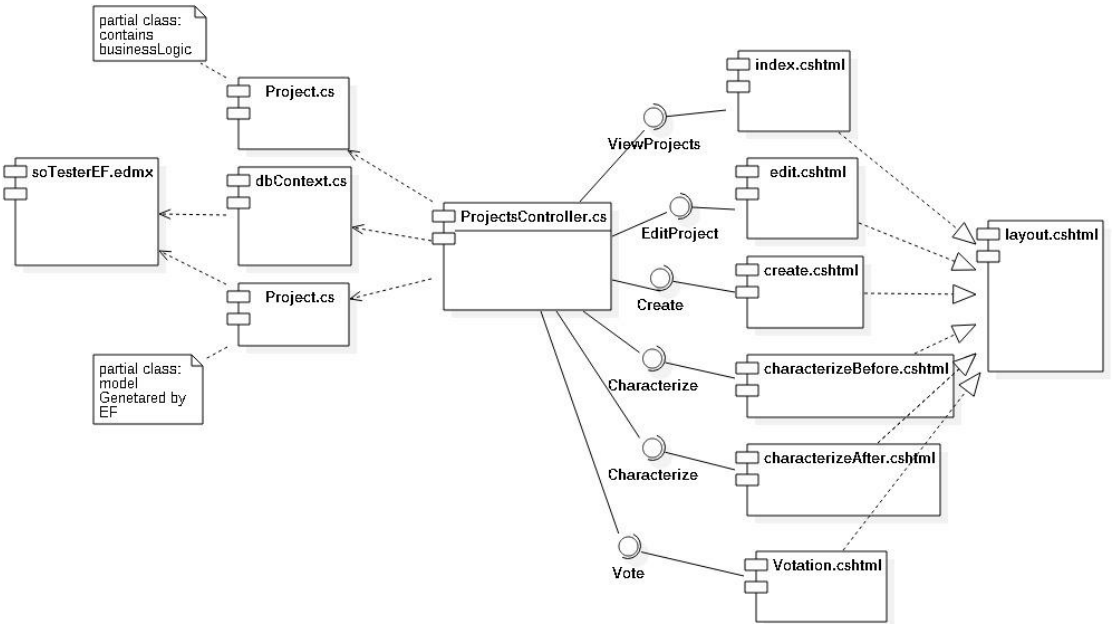

Figura 15 – **Diagrama de componentes - proyecto**.
Fuente. Elaboración propia

4.4.6. Vista física

Mediante el diagrama de despliegue de la Figura 16, se muestran los nodos físicos (servidores) y los artefactos de la solución que en ellos se despliegan.

Servidores de base de datos: según la infraestructura disponible para el despliegue de la aplicación y dado que la aplicación cuenta con dos bases de datos relacionales (base de datos del repositorio “SoTesttersDB” y la base de datos de seguridad “AccountDB”), cada una se encuentra en un servidor diferente (DatabaseServer01 y DatabaseServer02).

Servidor de aplicaciones: la aplicación se despliega en el servidor “Internet Information Services - IIS”, el cual a su vez tiene desplegado .net 4.0.

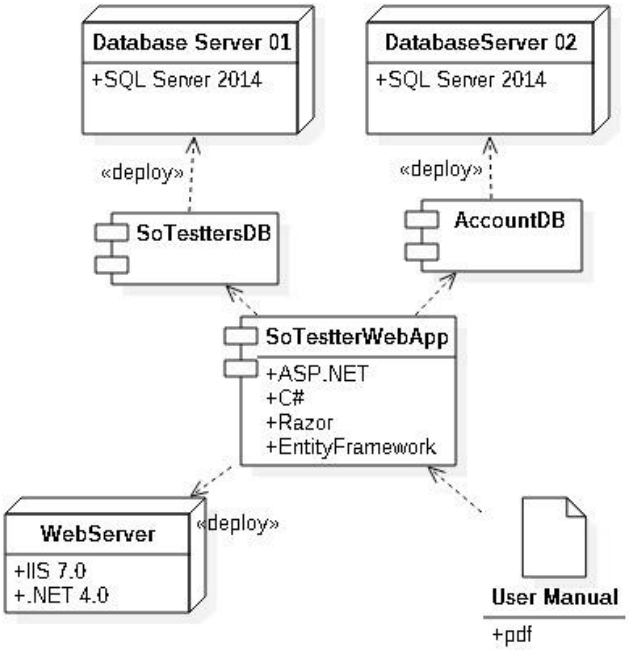

Figura 16 – **Diagrama de despliegue**.
Fuente. Elaboración propia

4.4.7. Modelo físico de la base de datos

En la Figura 17 y Figura 18, se muestran diagramas de base de datos relacional en los que se muestran las tablas principales con las que cuenta el repositorio para almacenar los proyectos, técnicas de testing, esquemas de caracterización y recomendaciones. Cada tabla del diagrama se corresponde con una clase del diagrama de clases ya presentado en la sección anterior.

Figura 17 - **Base de datos: vista de Proyecto**
Fuente. Elaboración propia

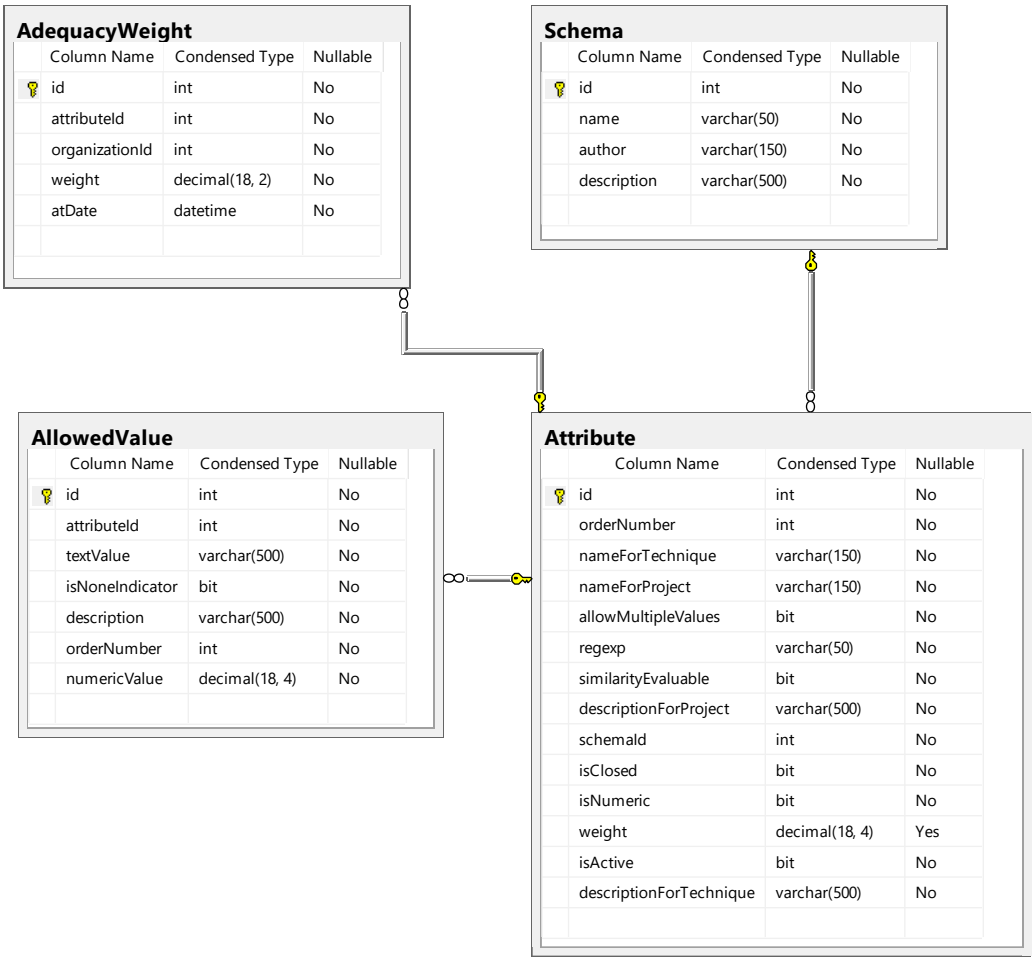

Figura 18 - **Base de datos: Esquema de caracterización**
Fuente. Elaboración propia

Interfaz gráfica de usuario. La aplicación cuenta con vistas web responsivas cuyo tamaño y elementos graficas se adecuan a la pantalla del dispositivo en el que se muestran. A continuación, se presentan las principales vistas.

Registro de una nueva organización. En esta vista, el usuario indica el nombre y descripción de la organización, así como la importancia de los atributos de similitud del esquema de caracterización.

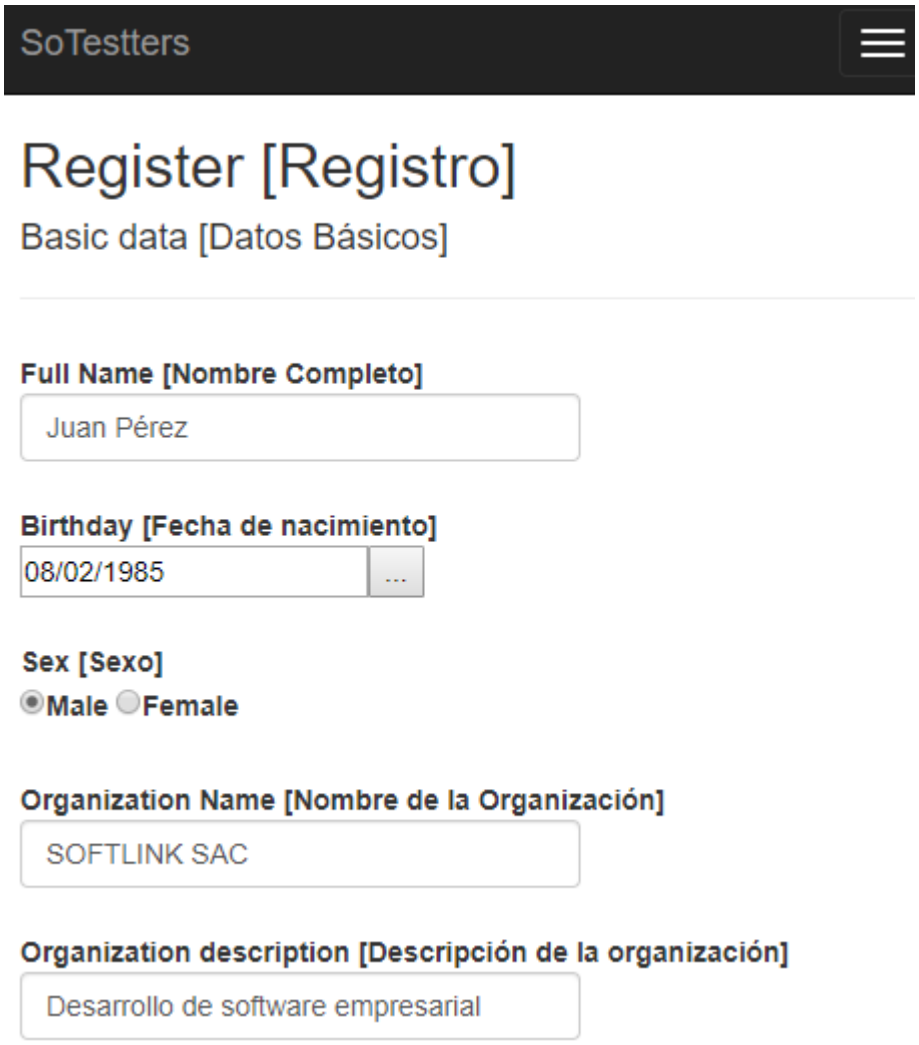

Figura 19 - **GUI: Registro de organización – parte 1**
Fuente. Elaboración propia

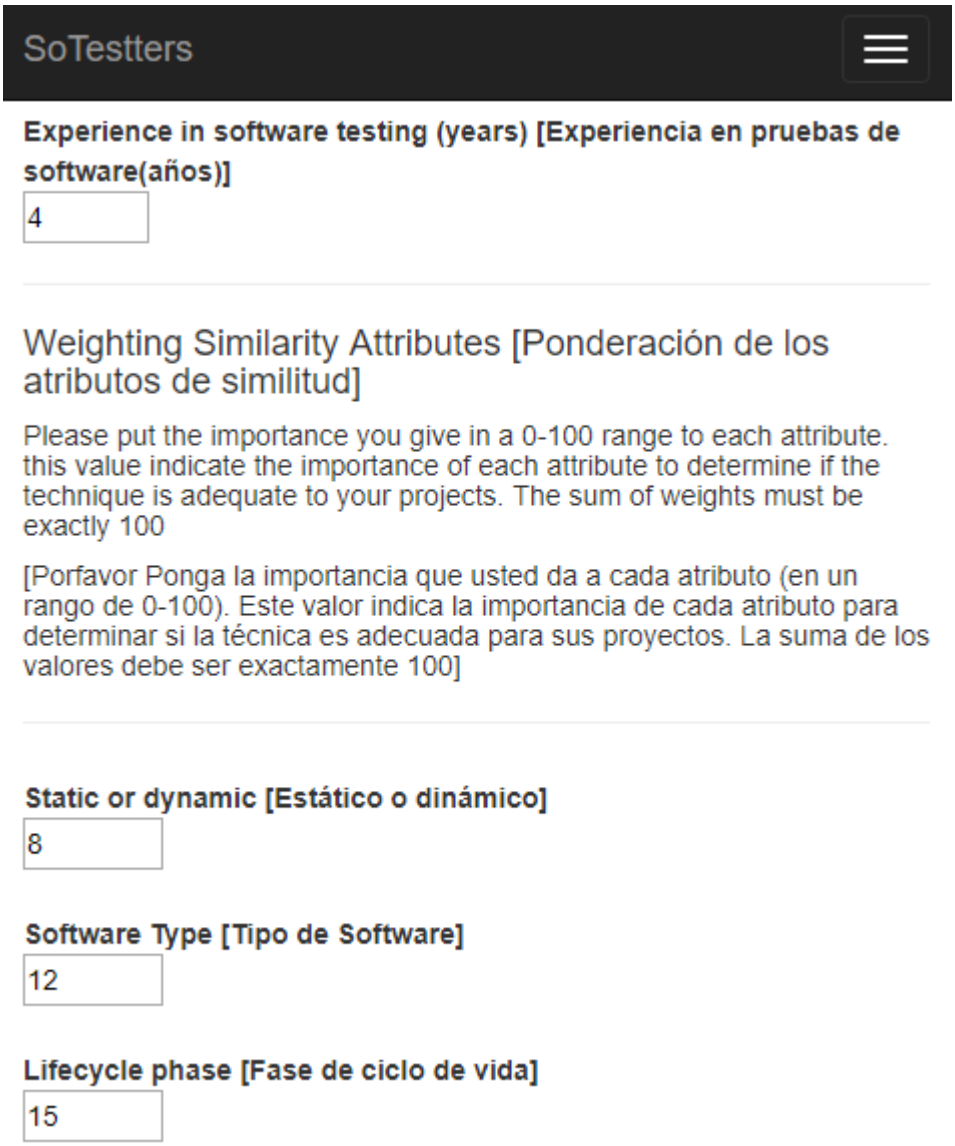

Figura 20 - **GUI: Registro de organización – parte 2**
Fuente. Elaboración propia

Creación de un nuevo proyecto. Formulario para el registro de los datos generales de un nuevo proyecto. La propiedad del proyecto se define internamente ligando el proyecto al usuario actualmente logueado.

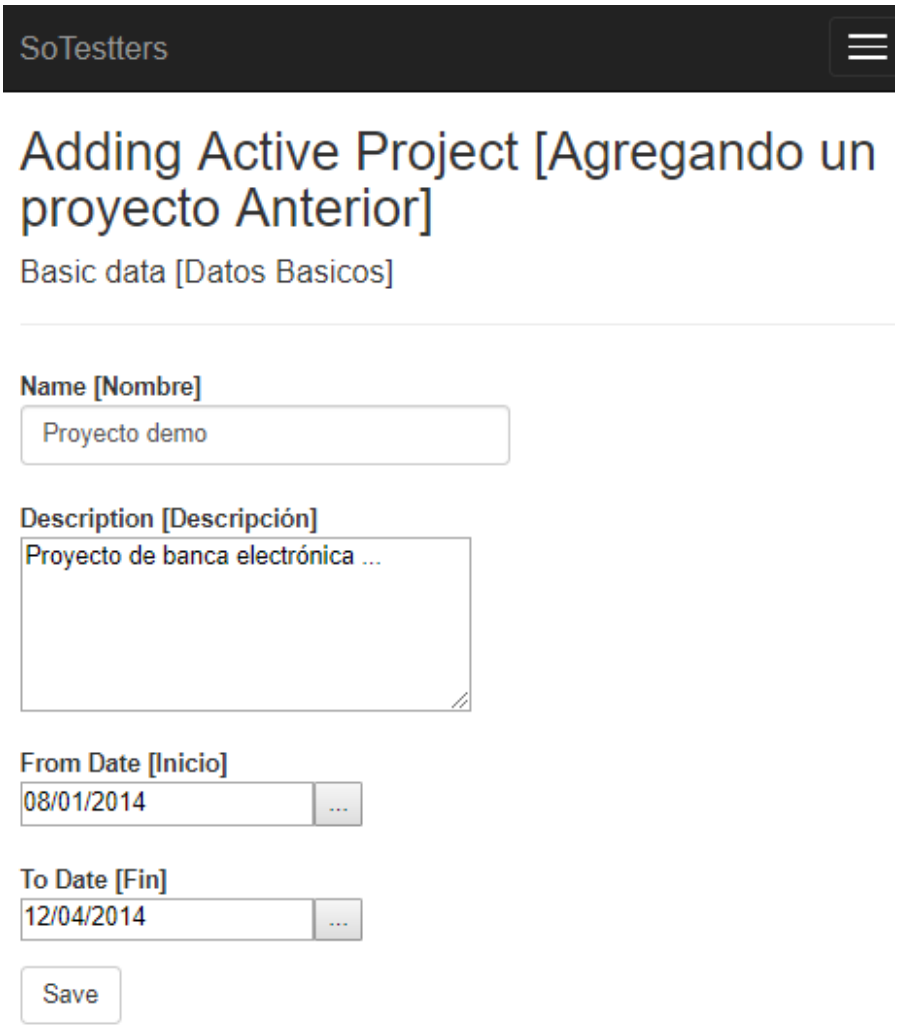

Figura 21 - **GUI: Crear proyecto**
Fuente. Elaboración propia

Listado de proyectos. El listado de proyecto funciona como un panel de control desde donde se pueden realizar todas las acciones posibles sobre un proyecto según el estado en el que se encuentra.

Figura 22 - **GUI: Listado de proyectos**
Fuente. Elaboración propia

Caracterización de proyecto: En esta vista, el propietario del proyecto indica las características de este, asignando valores a cada atributo de similitud del esquema de caracterización.

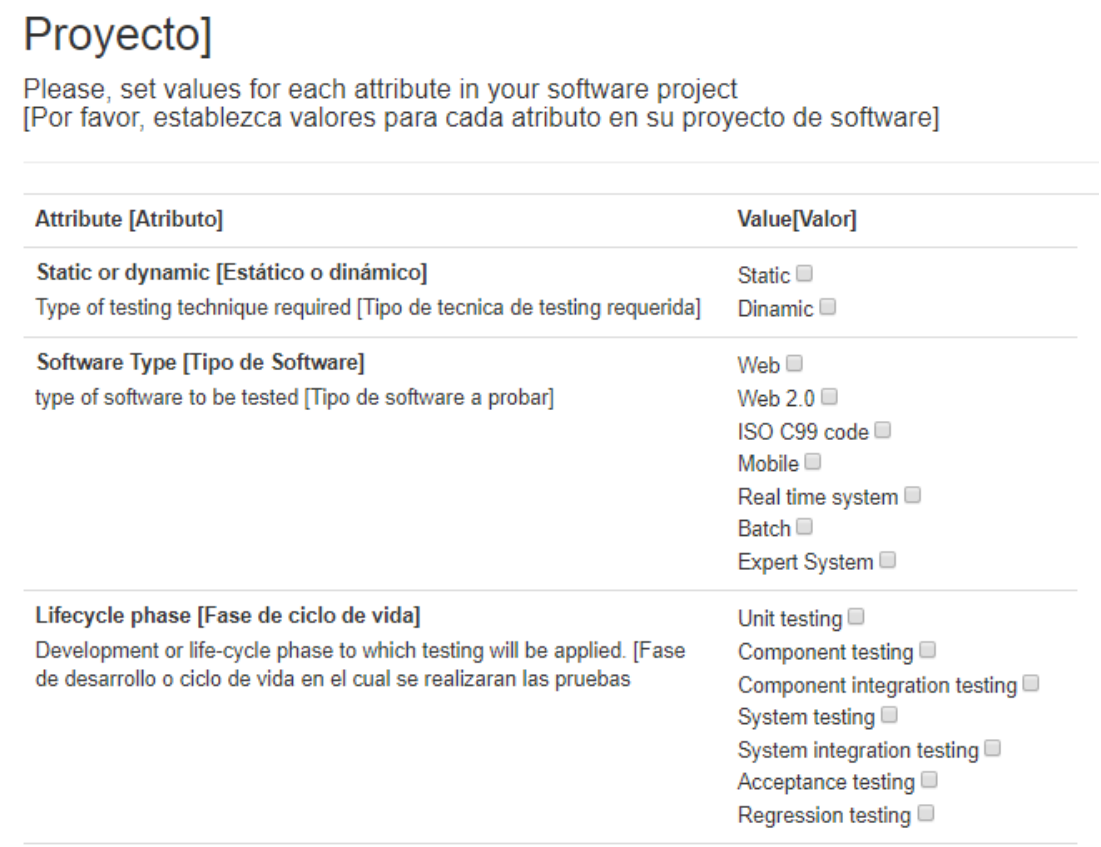

Figura 23 - **GUI: Caracterización de proyecto**
Fuente. Elaboración propia

Solicitar recomendación. En esta vista el usuario solicita la recomendación de técnicas de testing, para lo cual debe indicar la cantidad de técnicas que desea y luego las preferencias respecto a los atributos de desempeño, lo cual será empleado directamente para evaluar el desempeño de las técnicas instanciadas en proyectos similares al proyecto objetivo (Proyecto para el cual se desea obtener recomendaciones)

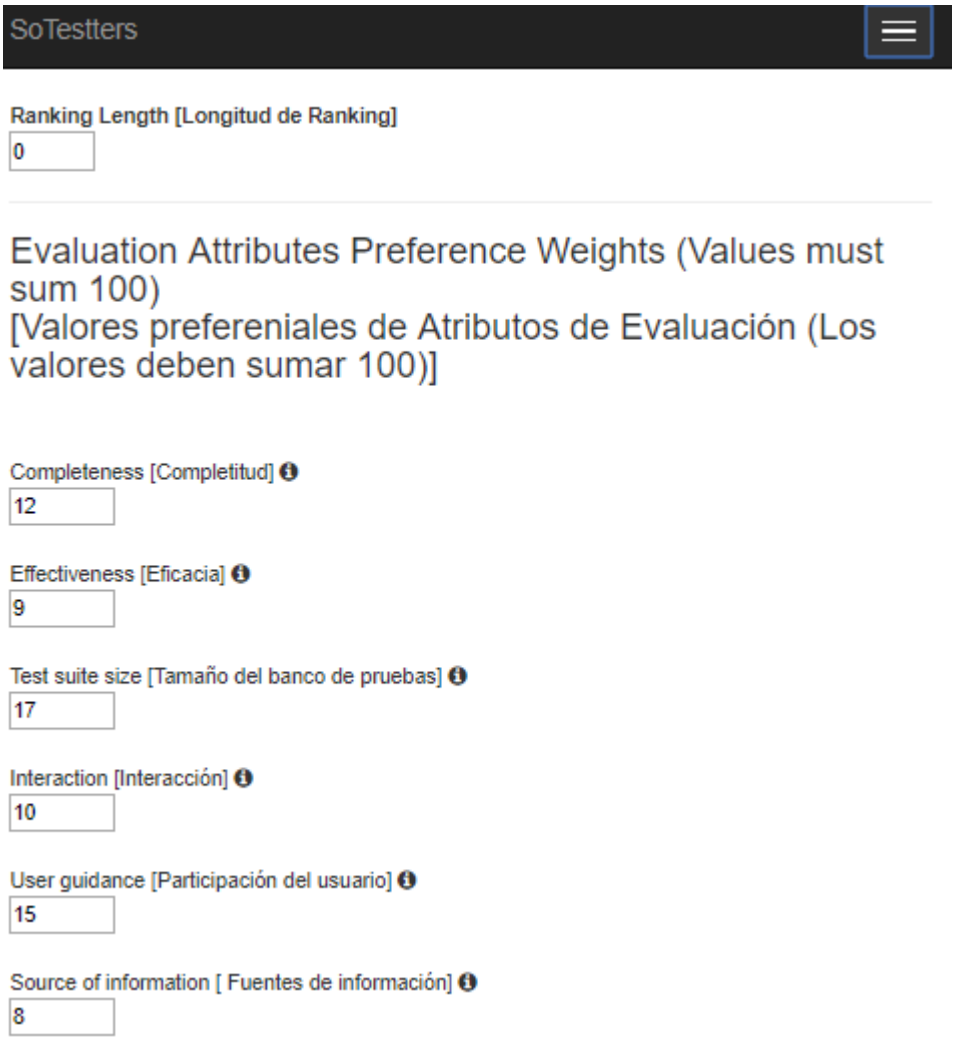

Figura 24 - **GUI: Solicitud de recomendación**
Fuente. Elaboración propia

Recomendación. En esta vista el sistema muestra las recomendaciones, en la vista simple se visualiza el ranking con el nombre de la técnica, la similitud y el puntaje obtenido en la evaluación de la técnica. En la vista detallada, se muestra los puntajes promedios de cada técnica por cada atributo de desempeño del esquema de caracterización.

Technique	Evaluation
R1: | Statement coverage	0.916
R2: | Function Coverage	0.893
R3: | State Transition Diagrams	0.780

Detail View

Results of evaluation by attribute and recommended technique (0-100 range) [Resultados de la evaluación por atributo y tecnica recomendada en un rango de 0 a 100]

Attribute	R1: Statement coverage	R2: Function Coverage	R3: State Transition Diagrams
Completeness [Completitud]	0.950	0.980	0.700
Effectiveness [Eficacia]	0.740	0.980	0.700
Test suite size [Tamaño del banco de pruebas]	0.990	0.800	0.650
Interaction [Interacción]	0.950	0.800	0.750
User guidance [Participación del usuario]	0.900	0.850	0.900
Source of information [Fuentes de información]	0.990	0.850	0.900
Comprehensibility [Comprensibilidad]	0.750	0.950	0.650
Subjective satisfaction [Satisfacción subjetiva]	0.950	0.950	0.850
Effort [Esfuerzo]	0.950	0.950	1.000

Figura 25 - **GUI: Ranking de técnicas recomendadas**
Fuente. Elaboración propia

Indicar técnicas instanciadas. El usuario accede a esta vista luego de culminar con el proceso de testing en su proyecto de software, aquí deberá indicar cual o cuales técnicas fueron instanciadas en su proyecto a partir de las recomendaciones realizadas por el sistema.

Selecting Instanced Techniques [(seleccionando técnicas instanciadas)]

Please, select all techniques you have applied in your Software Project
[Por favor, seleccione todas las técnicas que ha aplicado en su proyecto de software]

Techniques

Name [Nombre]	Description [Descripción]	Instanced? instancia?
Statement coverage	Structural White box testing : Control flow/ Coverage testing	☑
Function Coverage	Structural White box testing : Control flow/ Coverage testing	☑
State Transition Diagrams	blackbox testing	☐

Figura 26 - **GUI: Selección de técnicas instanciadas**
Fuente. Elaboración propia

Caracterización después de instanciación. En esta vista el usuario caracteriza las técnicas instanciadas en su proyecto, asignando un valor por cada atributo de desempeño.

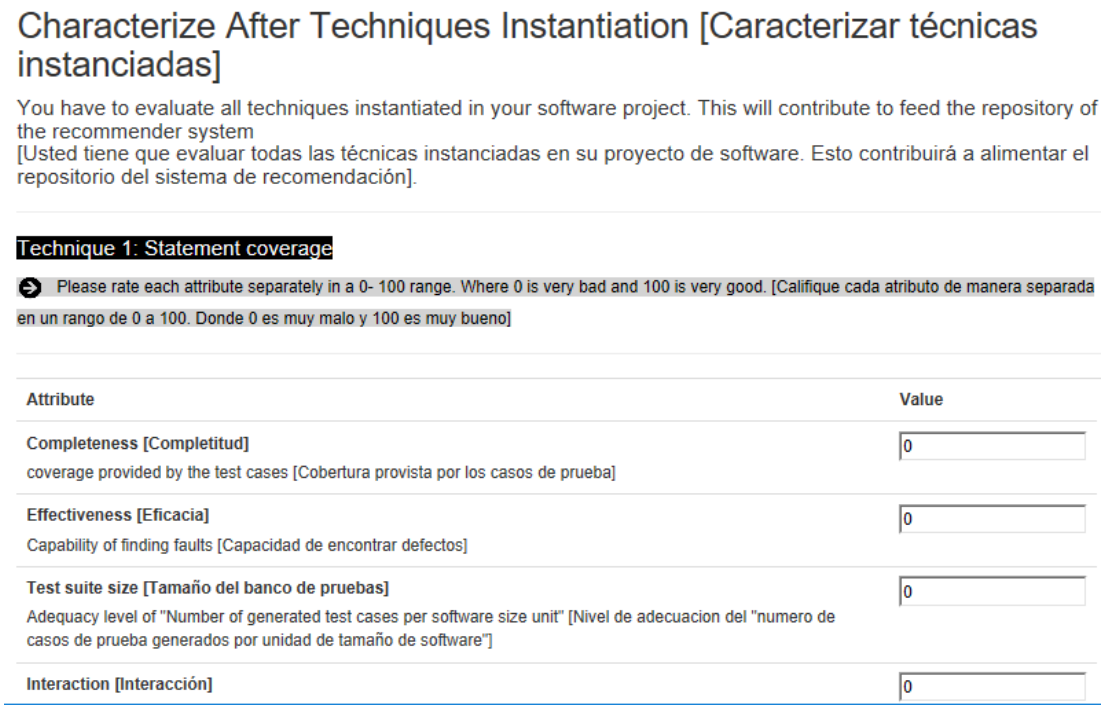

Figura 27 - GUI: **Caracterización de técnicas instanciadas**
Fuente. Elaboración propia

CAPÍTULO 5. RESULTADOS Y DISCUSION

5.1. Técnicas de testing

Con la finalidad de alimentar el repositorio, en primer lugar se registraron en el repositorio 23 técnicas de testing de software que se muestran en la Tabla 16, estas fueron recopiladas de dos trabajos de investigación (Jovanovic, 2009), (Nidhra, 2012).

Tabla 16 – **Técnicas de testing de software**

Nº	Técnica
1.	Equivalence Class Partitioning (particionamiento de clases equivalentes)
2.	Boundary Value Analysis (Análisis de valores límite)
3.	Decision Tables (Tablas de decisión)
4.	State Transition Diagrams (Diagramas de transición de estados)
5.	Orthogonal Arrays (Arreglos ortogonales)
6.	All Pairs Technique (Técnica de todos los pares)
7.	Fuzz testing (Pruebas fuzz)
8.	Cause-effect graphing (Gráficos de causa efecto)
9.	Desk checking (Comprobación de escritorio)
10.	Code walkthrough (Revisión de Código paso a paso)
11.	Formal Inspections (Inspecciones formales)
12.	Statement coverage (Cobertura de sentencia)
13.	Branch coverage (Cobertura de rama)
14.	Decision/Condition Coverage (Cobertura de decision/condición)
15.	Function Coverage (Cobertura de función)
16.	Flow Graph Notation (Notación de gráficos de flujo)
17.	Cyclomatic Complexity (Complejidad ciclomática)
18.	Deriving Test Cases (Casos de prueba derivados)
19.	Graph Matrices (Matrices gráficas)
20.	Simple Loops (Bucles simples)
21.	Nested loops (Bucles anidados)
22.	Concatenated loops (Bucles concatenados)
23.	Unstructured loop (Bucles no estructurados)

Fuente. Datos tomados de (Jovanovic, 2009) y de (Nidhra, 2012).

5.2. Proyectos históricos

En la Tabla 17 se muestra el perfil de los 12 profesionales participantes, en la que se percibe que todos los profesionales ocupan posiciones en el campo de testing de software y la experiencia media de los participantes fue de 6 años.

En total 12 profesionales aportaron un total de 26 proyectos históricos, de los cuales 11 profesionales aportaron 2 proyectos cada uno y un solo profesional aportó 4 proyectos.

Tabla 17 – **Perfil de profesionales que inicializaron el repositorio**

Profesional	Posición laboral	Años de experiencia en software testing
1	QA Manager	15
2	Lider Técnico Calidad	7
3	Consultor de procesos	7
4	QA	6
5	Analista de Calidad	6
6	Analista de Calidad	6
7	Analista de Certificaciòn_ Lider QA	6
8	Analista de Calidad	5
9	Analista de QA	5
10	Analista de Calidad	4
11	Analista de Calidad	3
12	Analista de Calidad	3

Fuente. Elaboración propia.

El listado de proyectos históricos, puede visualizarse en la *Tabla 18*.

Tabla 18 – **Proyectos históricos**

Nº	PROYECTO
1.	Mejoras al Proceso de Bloqueos
2.	TD CRM - Generación de Interfaces para Fuente $ST
3.	Sistema de cálculo de comisiones de agencias
4.	Gestión de Provisiones de Seguros
5.	PORTABILIDAD NUMERICA
6.	Implementación de BanTotal en Mi Banco
7.	Sistema NMIC
8.	Sistema de Gestión de desempeño
9.	Arial
10.	Módulo de compra y venta de CDS
11.	ADS Regulatorio
12.	SIGESTI
13.	Web de siniestros online
14.	Envío de Mensajes Telefonía
15.	Modificaciòn de Nòmina de Trabajador

16.	PROY: RO-ROPE
17.	GUIDEWIRE - MÓDULO BILLING CENTER
18.	Integración de Data Warehouse en Pacífico Seguros
19.	ATM - Banca - Comisiones Por Transacción
20.	Diferimiento de primas
21.	Separación OnLine de Diferidos
22.	PROYECTO CREO R1 - Módulo Pólicy
23.	Cobros por cargos automáticos
24.	Bapi Fino
25.	Flujo de impresión tarjeta de crédito
26.	MQ - BROKER BCP - DIRECTV (PAGOS)

Fuente. Elaboración propia.

5.3. Instanciación de técnicas

En la Tabla 19 se muestra la cuenta de instanciaciones por cada técnica del repositorio, ordenadas de manera descendente, observándose que:

- Del total de 23 técnicas registradas, 18 fueron instanciadas por lo menos una vez y 5 no fueron instanciadas.
- Se realizaron en total 71 instanciaciones.
- La técnica más instanciada es “Decision Tables” con 18 instanciaciones
- La media de instanciaciones por técnica es de 3.08

Tabla 19 – **Instanciación de técnicas en proyectos históricos**

Nº	Técnica	Cuenta de Instanciaciones
1	Decision Tables	18
2	Boundary Value Analysis	9
3	Equivalence Class Partitioning	7
4	Cause-effect graphing	7
5	State Transition Diagrams	4
6	All Pairs Technique	4
7	Formal Inspections	4
8	Statement coverage	3
9	Function Coverage	3
10	Branch coverage	2
11	Decision/Condition Coverage	2
12	Flow Graph Notation	2
13	Orthogonal Arrays	1
14	Fuzz testing	1
15	Desk checking	1
16	Code walkthrough	1
17	Deriving Test Cases	1
18	Graph Matrices	1
19	Simple Loops	0

20	Nested loops	0
21	Concatenated loops	0
22	Unstructured loop	0
23	Cyclomatic Complexity	0

Fuente. Elaboración propia.

5.4. Proyectos Objetivo

Se consiguió un total de 11 proyectos objetivo, los cuales se listan en la Tabla 20. Estos fueron caracterizados por los miembros del equipo de proyecto, quienes además indicaron sus preferencias respecto a la recomendación de técnicas de testing. En el Apéndice 1 se muestra la caracterización de cada proyecto objetivo y sus preferencias de recomendación.

Tabla 20 – **Proyectos objetivo**

N°	Nombre
1	SIGESCOM
2	GLOBALNET
3	SISAP LAB
4	SISTEMA VIRTUAL DE AUTOEVALUACION
5	Gestion de almacen
6	Gestion produccion operarios
7	INTEGRACION JUEGOS DE AZAR
8	control de tareo
9	Sotester
10	Detracciones
11	Control de materiales en producción

Fuente. Elaboración propia.

5.5. Pruebas de Hipótesis

Se realizó tres pruebas: dos para las hipótesis específicas y una para la hipótesis general. En todos los casos se contempló dos variables estadísticas, “Experts” y “System” tal como se muestra en la *Figura 29*.

```
# Diferencias entre las recomendaciones de los expertos
Experts <- c(1,1,1,0,0,0,1,1,1,0,0,2,1,2,1,1,1,1,1,0,0,0,1,1,2,1,1,0,1,0,0,1,0)
# Diferencias entre las recomendaciones del Sistema SOTESTER y los expertos humanos
System <- c(0,0,0,1,2,0,0,1,0,0,1,1,0,0,1,1,0,1,0,1,1,1,0,0,0,1,1,0,0,2,2,1,0)
```

Figura 28 – **Variables estadísticas para las pruebas de hipótesis**
Fuente. Elaboración propia

5.5.1. Hipótesis especifica 1 (H_{e1})

La primera hipótesis específica es “La coincidencia de los expertos respecto a las técnicas de testing de software que recomiendan, será mayor a 1 en un rango de 0 a 3”. Al respecto, en los resultados, según la Tabla 21, se muestra las coincidencias de las recomendaciones realizadas por los expertos. Estas diferencias toman valores entre cero y tres, donde cero indica que no existen coincidencias entre las técnicas recomendadas y tres significa que recomendaron exactamente las mismas técnicas.

En el Apéndice 4, se presentan las técnicas recomendadas por los expertos para cada proyecto objetivo.

Tabla 21. **Coincidencias entre las recomendaciones de los expertos**

PO	EXP1 vs EXP2	EXP1 vs EXP3	EXP2 vs EXP3
1	1	2	1
2	1	1	1
3	1	2	2
4	0	1	1
5	0	1	1
6	0	1	0
7	1	1	1
8	1	1	0
9	1	0	0
10	0	0	1
11	0	0	0

Fuente. Elaboración propia.

En la Figura 29, se muestra la cuenta por cada factor de la variable estadística, comprobándose que, de un total de 33 observaciones, 12 tienen cero coincidencias, 18 tienen una coincidencia y tres observaciones tienen dos coincidencias.

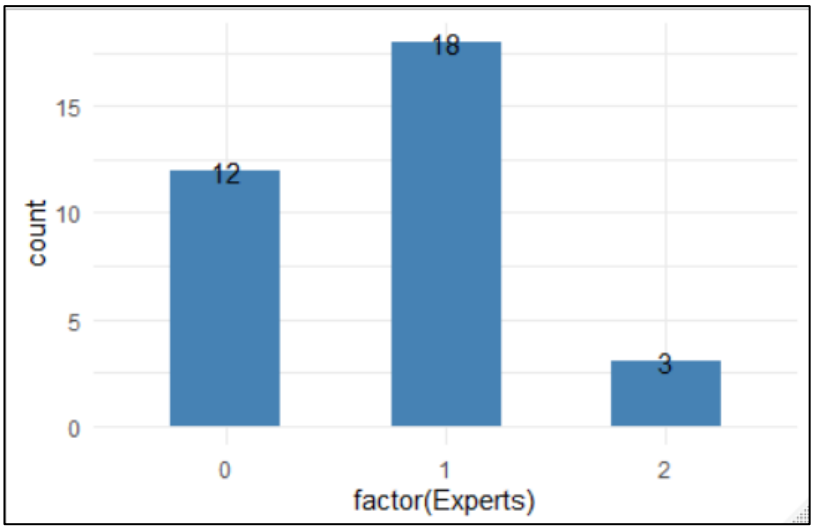

Figura 29 - **Coincidencias en las técnicas recomendadas por los expertos**
Fuente. Elaboración propia

a) Selección del estadístico de prueba

Se utiliza la media como medida descriptiva de los resultados,

b) Planteamiento de la hipótesis estadística

$$H_0: 1 \leq \mu 1 \leq 3, H_1: \mu 1 < 1$$

c) Regla de decisión

Si $1 \leq \mu 1 \leq 3$, se acepta H_0, si no, se acepta H_1

d) Cálculo estadístico y toma de decisión

En la Figura 30, se muestra la estadística descriptiva de las coincidencias entre expertos, obteniéndose un valor mínimo de cero y máximo de dos; además se obtuvo una media de 0.73 y una mediana de 1.00.

En base a estos resultados, se acepta la hipótesis alterna: La media de las coincidencias entre expertos de técnicas de testing de software recomendadas por los expertos es menor a 1.

nbr.val	nbr.null	nbr.na	min	max
33.0000000	12.0000000	0.0000000	0.0000000	2.0000000
range	sum	median	mean	SE.mean
2.0000000	24.0000000	1.0000000	0.7272727	0.1089962
CI.mean.0.95	var	std.dev	coef.var	
0.2220179	0.3920455	0.6261353	0.8609361	

Figura 30 – **Medidas descriptivas de las coincidencias en las técnicas recomendadas por los expertos**
Fuente. Elaboración propia mediante software R Studio

5.5.2. Hipótesis especifica 2 (H_{e2})

La segunda hipótesis específica es "La coincidencia del método propuesto SOTESTER con los expertos respecto a las técnicas de testing de software que recomiendan será mayor a 1 en un rango de 0 a 3" Al respecto, en los resultados, según la Tabla 22, se muestra las coincidencias entre el método SOTESTER y los expertos respecto a las técnicas recomendadas para cada uno de los once proyectos objetivo.

En el Apéndice 4, se presentan las técnicas recomendadas por SOTESTER para cada proyecto objetivo.

Tabla 22 **– Coincidencias entre recomendaciones de SOTESTER y expertos**

PO	SOT vs EXP1	SOT vs EXP2	SOT vs EXP3
1	0	1	0
2	0	0	0
3	0	0	0
4	1	1	1
5	2	1	1
6	0	0	0
7	0	1	0
8	1	0	2
9	0	1	2
10	0	1	1
11	1	1	0

Fuente. Elaboración propia.

En la *Figura 31*, se muestra la cuenta por cada factor de la variable estadística, comprobándose que, de un total de 33 observaciones, 17 tienen cero coincidencias, 13 tienen una coincidencia y tres observaciones tienen dos coincidencias.

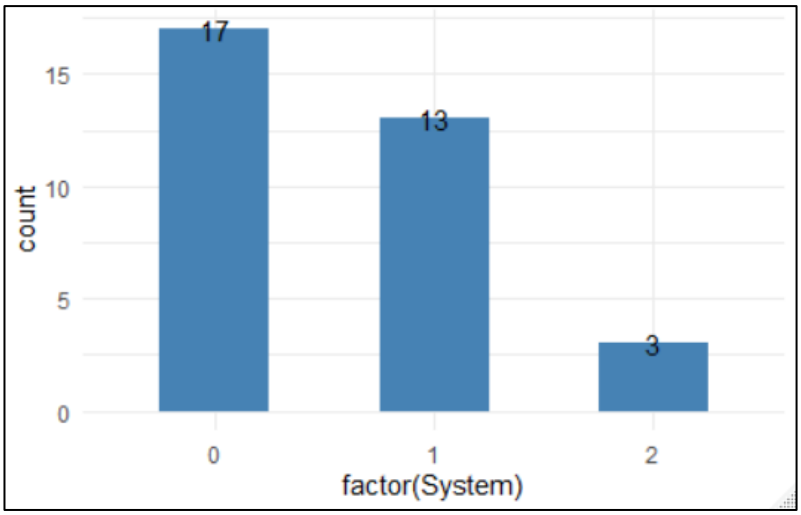

Figura 31 - **Coincidencias en las técnicas recomendadas por SOTESTER y expertos**
Fuente. Elaboración propia

a) Selección del estadístico de prueba

Se utiliza la media como medida descriptiva de los resultados,

b) Planteamiento de la hipótesis estadística

$$H_0: 1 \leq \mu 1 \leq 3, H_1: \mu 1 < 1$$

c) Regla de decisión

Si $1 \leq \mu 1 \leq 3$, se acepta H_0, si no, se acepta H_1

d) Cálculo estadístico y toma de decisión

En la *Figura 32*, se muestra la estadística descriptiva de las coincidencias entre expertos, obteniéndose un valor mínimo de cero y máximo de dos; además se obtuvo una media de 0.58 y una mediana de 0.00. En base a estos resultados, se acepta la hipótesis alterna: La media de las coincidencias de técnicas de testing de software recomendadas por SOTESTER y los expertos es menor a 1.

nbr.val	nbr.null	nbr.na	min	max
33.0000000	17.0000000	0.0000000	0.0000000	2.0000000
range	sum	median	mean	SE.mean
2.0000000	19.0000000	0.0000000	0.5757576	0.1153905
CI.mean.0.95	var	std.dev	coef.var	
0.2350428	0.4393939	0.6628680	1.1512970	

Figura 32 – **Medidas descriptivas de las coincidencias en las técnicas recomendadas por SOTESTER y los expertos**
Fuente. Elaboración propia mediante software R Studio

5.5.3. Hipótesis general

La prueba de hipótesis general, se realizó utilizando análisis inferencial frecuentista mediante la comparación de medias, haciendo uso del software estadístico R y R Studio.

a) Selección del estadístico de prueba

Debido a que se requiere comparar dos grupos con muestras pareadas y además se trata de una variable discreta, se utilizó la prueba de rangos con signo de Wilcoxon teniendo en cuenta la modificación de Pratt en caso de que existan ceros (Fay y Malinovsky, 2018, p. 26).

b) Planteamiento de hipótesis estadística

$$H_0: median(Experts - System) = 0, H_1: median(Experts - System) \neq 0$$

c) Regla de decisión

Si $p - value > 0.05$, se acepta H_0, si no, se acepta H_1

d) Cálculo estadístico y toma de decisión

En la *Figura 33*, se muestra un diagrama de frecuencias a modo de comparación de las coincidencias por un lado entre expertos (Experts) y por otro lado, SOTESTER versus expertos (System), pudiéndose apreciar frecuencias aproximadamente parecidas por cada factor de coincidencia (0,1,2).

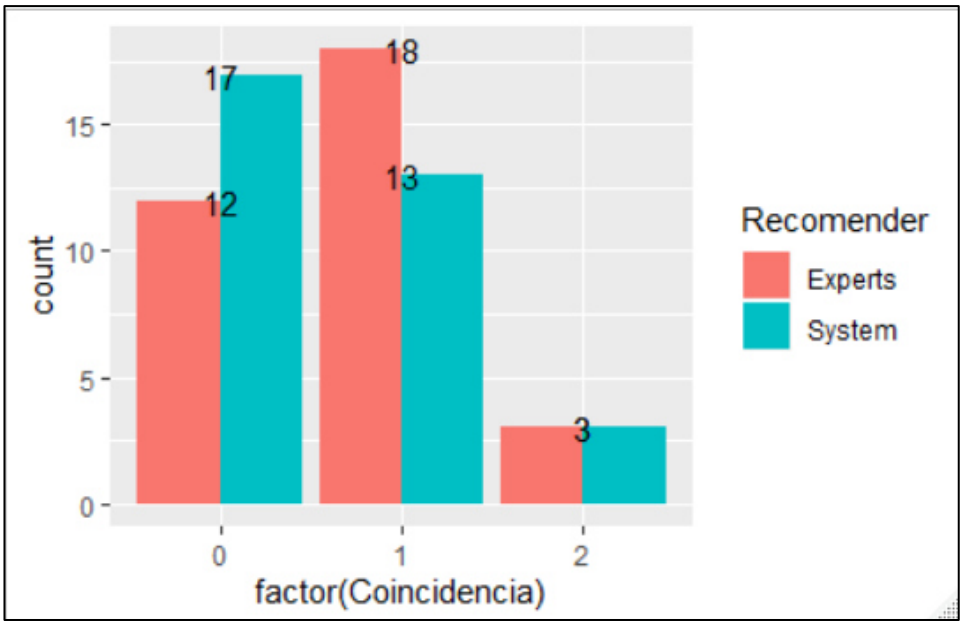

Figura 33 – ***Diagrama de frecuencias de las coincidencias – Expertos(Experts) vs. SOTESTER(System)***
Fuente. Elaboración propia

En la *Figura 34*, se muestra los resultados de la prueba estadística, comprobándose que el p-value es 0.3498, resultando mayor a 0.05, por lo que se acepta la hipótesis estadística H_0 la cual asume que la mediana de las diferencias es igual a cero; por lo tanto, se acepta la Hipótesis general de investigación : La Calidad de las recomendaciones de técnicas de testing de software en base al método de recomendación de técnicas de testing SOTESTER, es buena.

```
        Exact Wilcoxon Signed-Rank Test (with Pratt modification if
        zeros)

data:  Experts minus System
p-value = 0.3498
alternative hypothesis: true median is not equal to 0
95 percent confidence interval:
 0.0 0.5
sample estimates:
median (Hodges-Lehmann estimator)
                     6.103516e-05
```

Figura 34 – ***Prueba de hipótesis: Mediana de las diferencias entre Expertos vs. Expertos y SOTESTER vs. Expertos***
Fuente. Elaboración propia

CONCLUSIONES

Se ha propuesto el método SOTESTER para recomendar técnicas de testing de software mediante un enfoque colaborativo basado en contenido, a partir de un repositorio dinámico en base al cual se ofrecen recomendaciones para proyectos objetivos, los cuales al cerrarse, se convierten en proyectos históricos para retroalimentar el repositorio y por ende ayudar a mejorar la calidad de las recomendaciones, concluyendo que teóricamente existe un mecanismo para recomendar técnicas de testing de manera colaborativa y con catálogo dinámico.

Se ha logrado construir e implementar una aplicación Web escalable que implementa el método SOTESTER, inicializando el repositorio de SOTESTER con un total de 23 técnicas de testing de software y 26 proyectos históricos, obteniendo un total de 71 instanciaciones de técnicas de testing. También se ha logrado obtener recomendaciones de técnicas de testing mediante el uso del método SOTESTER para once proyectos de software a los que se denomina proyectos objetivo; lo cual permite concluir que es posible aplicar el método SOTESTER de manera práctica para soportar trabajos de investigación y actividades reales en la industria.

Se logró medir la coincidencia entre expertos respecto a las técnicas de testing de software que recomiendan, encontrando que de 3 técnicas que recomiendan, la coincidencia media es de 0.73 técnicas y en términos prácticos coinciden en una técnica, lo cual permite concluir que existe un mínimo consenso entre expertos al momento de recomendar técnicas de testing para un proyecto objetivo dado. Esto podría estar indicando que se deben contemplar características más específicas acerca de los proyectos de software a probar.

Se logró medir la coincidencia del método propuesto SOTESTER con los expertos respecto a las técnicas de testing de software que recomiendan, encontrando que de 3 técnicas que recomiendan, la coincidencia media es de 0.58 técnicas y en términos prácticos coinciden en una técnica, lo cual permite concluir que existe un mínimo consenso entre el método propuesto SOTESTER y los expertos al momento de recomendar técnicas de testing.

Se logró medir la calidad de las recomendaciones realizadas por el método SOTESTER desde una perspectiva de la exactitud, mediante la comparación de (1) coincidencias entre expertos respecto a (2) las coincidencias entre SOTESTER y expertos respecto a las técnicas de testing de software que recomiendan para los proyectos objetivo, encontrándose que no hay diferencias significativas entre tales coincidencias, concluyendo que las recomendaciones realizadas por el método SOTESTER son de buena calidad y que SOTESTER puede ofrecer recomendaciones de técnicas de testing de software con resultados similares a los de un experto.

TRABAJOS FUTUROS

Se podría complementar la validación del método obteniendo recomendaciones para otros proyectos objetivos, enfocándose también en el performance del método con más proyectos históricos que aporten mayor diversidad de ítems a recomendar, probando distintos algoritmos de recomendación manteniendo el enfoque colaborativo.

El análisis de combinación es un tema importante que debería ensayarse para ofrecer mejores recomendaciones, de modo que se facilite aún más la decisión del usuario respecto a la instanciación de técnicas de testing en su proyecto de software teniendo en cuenta que la instanciación de cada técnica implica el uso de ciertos recursos.

Una vez usado el sistema de recomendación con una mayor cantidad de proyectos y recomendaciones, aprovechando que el sistema memoriza las recomendaciones realizadas, se sugiere obtener recomendaciones teniendo en cuenta el feedback de las recomendaciones pasadas como indicador del nivel de confianza y posteriormente realizar una evaluación desde la perspectiva del usuario.

REFERENCIAS

Armentano, M. G., Christensen, I., y Schiaffino, S. (2015). Applying the Technology Acceptance Model to Evaluation of Recommender Systems, (51), 73-79.

Arora, A., y Sinha, M. (2012). Web Application Testing: A Review on Techniques, Tools and State of Art. *International Journal of Scientific {\&} Engineering Research*, *3*(2), 1-6. Recuperado a partir de http://www.ijser.org/researchpaper/Web-Application-Testing-A-Review-on-Techniques-Tools-and-State-of-Art.pdf

Avazpour, I., Pitakrat, T., Grunske, L., y Grundy, J. (2014). Dimensions and metrics for evaluating recommendation systems. *Recommendation Systems in Software Engineering*, 245-273. https://doi.org/10.1007/978-3-642-45135-5_10

Bobadilla, J., Ortega, F., Hernando, A., y Gutiérrez, A. (2013). Recommender systems survey. *Knowledge-Based Systems*, *46*, 109-132. https://doi.org/10.1016/j.knosys.2013.03.012

Brosse, E., Vos, T., y Condori, N. (2014). Evaluating the FITTEST Automated Testing Tools in SOFTEAM : an Industrial Case Study, (May).

Chillarege, R., Bhandari, I. S., Chaar, J. K., Halliday, M. J., Ray, B. K., y Moebus, D. S. (1992). Orthogonal Defect Classification—A Concept for In-Process Measurements. *IEEE Transactions on Software Engineering*, *18*(11), 943-956. https://doi.org/10.1109/32.177364

Condori, N., Kruse, P. M., Vos, T. E. J., Brosse, E., y Bagnato, A. (2014). Combinatorial testing in an industrial environment - Analyzing the applicability of a tool. *Proceedings - 2014 9th International Conference on the Quality of Information and Communications Technology, QUATIC 2014*, 210-215. https://doi.org/10.1109/QUATIC.2014.36

Cotroneo, D., Pietrantuono, R., y Russo, S. (2013). Testing techniques selection based on ODC fault types and software metrics. *Journal of Systems and Software*, *86*(6), 1613-1637. https://doi.org/10.1016/j.jss.2013.02.020

Dias, A. C., y Travassos, G. H. (2009). Model-based testing approaches selection for software projects. *Information and Software Technology*, *51*(11), 1487-1504. https://doi.org/10.1016/j.infsof.2009.06.010

Dias, A. C., y Travassos, G. H. (2014). Supporting the Combined Selection of Model-Based Testing Techniques. *IEEE Transactions on Software Engineering*, *40*(10), 1025-1041. https://doi.org/10.1109/TSE.2014.2312915

Eldh, S., Hansson, H., Punnekkat, S., Pettersson, A., y Sundmark, D. (2006). A framework for comparing efficiency, effectiveness and applicability of software

testing techniques. *Proceedings - Testing: Academic and Industrial Conference - Practice and Research Techniques, TAIC PART 2006*, 159-170. https://doi.org/10.1109/TAIC-PART.2006.1

Engström, E., Runeson, P., y Skoglund, M. (2010). A systematic review on regression test selection techniques. *Information and Software Technology*, *52*(1), 1-35. https://doi.org/10.1016/j.infsof.2009.07.001

Farooq, S. U., y Quadri, S. M. K. (2013). Empirical Evaluation of Software Testing Techniques – Need , Issues and Mitigation. *Software Engineering : an International Journal (SEIJ)*, *3*(1), 41-51. https://doi.org/10.1145/2593690.2593693

Fay, M. P., y Malinovsky, Y. (2018). Confidence intervals of the Mann-Whitney parameter that are compatible with the Wilcoxon-Mann-Whitney test. *Statistics in Medicine*, *37*(27), 3991-4006. https://doi.org/10.1002/sim.7890

Felfernig, A., Jeran, M., y Ninaus, G. (2013). Toward the Next Generation of Recommender Systems: Applications and Research Challenges. *Multimedia Services in Intelligent Environments*, pp 81-98. https://doi.org/10.1007/978-3-319-00372-6_5

Jovanovic, I. (2009). Software Testing Methods and Techniques. *The IPSI BgD Transactions on Internet Research*, *5*(1), 30-41. Recuperado a partir de http://www.internetjournals.net/journals/tir/2009/January/Full Journal.pdf#page=31

Khan, M. E. (2010). Different forms of software testing techniques for finding errors. *International Journal of Computer Science Issues*, *7*(3), 11-16. https://doi.org/10.1.1.402.9250

Lawanna, A. (2014). The Improvement of Test Case Selection for the Process Software Maintenance The Improvement of Test Case Selection for the Process Software Maintenance. *INFORMATION TECHNOLOGY JOURNAL*, *10*(July).

Luo, L. (2001). Software testing techniques. *Institute for software research international Carnegie mellon university Pittsburgh, PA*, *15232*(1-19), 19.

Muthusamy, T., y Seetharaman, K. (2014). Effectiveness of Test Case Prioritization Techniques based on Regression Testing. *International Journl of Software Engineering & Application*, *5*(6), 113-123.

Nidhra, S. (2012). Black Box and White Box Testing Techniques - A Literature Review. *International Journal of Embedded Systems and Applications*, *2*(2), 29-50. https://doi.org/10.5121/ijesa.2012.2204

Ohsugi, N, Tsunoda, M, Monden, y A. (2004). Effort estimation based on Collaborative Filtering. *the 5th International Conference on Product Focused Software Process Improvement*, *3009*, 274-286. Recuperado a partir de http://apps.isiknowledge.com/full_record.do?product=UA&search_mode=RelatedRecords&qid=15&SID=1FywOeZlqK6hcvDM64B&page=2&doc=17

Pargament, K. I., Koenig, H. G., y Perez, L. M. (2000). Javawock : A Java Class Recommender System Based on Collaborative Filtering. *Journal of Clinical Psychology*, *56*(July), 519-543. Recuperado a partir de http://www.ncbi.nlm.nih.gov/pubmed/10775045

Pilar, M., Simmonds, J., y Astudillo, H. (2014). Semi-automated tool recommender for software development processes. *Electronic Notes in Theoretical Computer*

Science, *302*, 95-109. https://doi.org/10.1016/j.entcs.2014.01.022

Pu, P., y Chen, L. (2011). A User - Centric Evaluation Framework for Recommender Systems. *Proceedings of the 5th ACM conference on Recommender systems - RecSys '11*, 157-164. https://doi.org/10.1145/2043932.2043962

Resnick, P., Iacovou, N., Suchak, M., Bergstrom, P., y Riedl, J. (1994). GroupLens: An Open Architecture for Collaborative Filtering of Netnews. *Proceedings of the 1994 ACM conference on Computer supported cooperative work - CSCW '94*, 175-186. https://doi.org/10.1145/192844.192905

Robillard, M. P., Maalej, W., Walker, R. J., y Zimmermann, T. (2014). *Recommendation Systems in Software Engineering*. https://doi.org/10.1007/978-3-642-45135-5

Said, A., Fields, B., Jain, B. J., y Albayrak, S. (2013). User-centric Evaluation of a K-furthest Neighbor Collaborative Filtering Recommender Algorithm. *Proceedings of the 2013 Conference on Computer Supported Cooperative Work*, 1399-1408. https://doi.org/10.1145/2441776.2441933

Vegas, S. (2002). Characterization Schema for Selecting Software Testing Techniques. *Facultad. de Informática*, (February).

Vegas, S., y Basili, V. R. (2005). A Characterization Schema for Software Testing Techniques. *Empirical Software Engineering*, *10*(4), 437-466. https://doi.org/10.1007/s10664-005-3862-1

Vos, T., Mar, B., Panach, I., Baars, A., y Ayala, C. (2011). Evaluating Software Testing Techniques and Tools, 5-7.

Vos, T., Marin, B., Jose Escalona, M., Marchetto, A., Maríint, B., Escalona, M. J., y Marchetto, A. (2012). A Methodological Framework for Evaluating Software Testing Techniques and Tools. *12th International Conference on Quality Software, QSIC 2012*, *Xi'an*, *Sha*, 230-239. https://doi.org/10.1109/QSIC.2012.16

Zaidan, A. A., Zaidan, B. B., Al-Haiqi, A., Kiah, M. L. M., Hussain, M., y Abdulnabi, M. (2015). Evaluation and selection of open-source EMR software packages based on integrated AHP and TOPSIS. *Journal of Biomedical Informatics*, *53*, 390-404. https://doi.org/10.1016/j.jbi.2014.11.012

APENDICES

Apéndice 1 – FICHA DE RECOMENDACIÓN PARA EXPERTOS

I. DESCRIPCIÓN Y CARACTERIZACIÓN DEL PROYECTO

SIGESCOM: Sistema de gestión comercial para mypes: gestión de ventas, compras, clientes, proveedores, facturación, gastos y costos.

Atributo	Descripción	Valor
Estático o dinámico	Tipo de técnica de testing requerida	Dinámico
Tipo de Software	Tipo de software a probar	Web
Fase de ciclo de vida	Fase de desarrollo o ciclo de vida en el cual se realizaran las pruebas	Pruebas Unitarias Pruebas de componentes Pruebas del sistema
Escalabilidad	Tamaño del sistema a ser probado (LoC)	>= 3001
Entorno: lenguaje de programación	Lenguaje de programación usado para el desarrollo de software	C#
Entrada	¿Qué entradas están disponibles para el proceso de testing?	Código fuente Programa ejecutable
Conocimiento	¿Qué conocimientos tiene el equipo de pruebas?	Dominio de la aplicación Conocimiento acerca de cómo identificar el estado del sistema a probar Paradigma orientado a objetos Programación Trabajo en equipo Liderazgo
Experiencia	¿Qué experiencia tiene el equipo de pruebas?	Web testing
Salida	Salidas que el equipo espera producto de la aplicación de las técnicas de pruebas	Casos de prueba Defectos
Tipos de defectos	Principales tipos de defectos que se quiere detectar	Verificación o Checking Algoritmos Sincronización/Serialización
Aplicabilidad en tareas	Actividades de pruebas que se quiere realizar	Pruebas unitarias Pruebas del sistema
Obtención de herramienta	Tipo de licenciamiento que prefiere para la herramienta de pruebas	Comercial

II. PREFERENCIAS DEL EQUIPO DE PROYECTO RESPECTO AL DESEMPEÑO DE LAS TECNICAS A RECOMENDAR

A continuación, se indica los pesos asignados por el equipo de proyecto a los atributos de desempeño de las técnicas de testing que se le serán recomendadas. Cuanto mayor sea el valor asignado a un atributo, más relevancia tendrá tal atributo en el análisis de desempeño de las técnicas a recomendar, por lo que se deberá recomendar técnicas con mejor desempeño en los atributos más relevantes. Mientras que el valor cero indica que el atributo es irrelevante. La suma de pesos es 100.

Atributo	Descripción	Peso
Completitud	Cobertura provista por los casos de prueba	15
Eficacia	Capacidad de encontrar defectos	20
Tamaño del banco de pruebas	Nivel de adecuación del número de casos de prueba generados por unidad de tamaño de software	12
Interacción	Cuán adecuados son los modos de interacción; (Por ejemplo: navegación, consultas, refinamiento sucesivo, etc.) soportados por la técnica	5

Guía o participación del usuario	Cuan apropiada es la guía o participación que la técnica requiere del usuario. Por ejemplo: Nada apropiada (la técnica es completamente manual), evaluación manual de las salidas, selección de entradas apropiadas, definición de patrones, filtrado, etc.	5
Fuentes de información	Disponibilidad de fuentes de información acerca de la técnica	8
Comprensibilidad	Si la técnica es o no es fácil de entender	10
Satisfacción subjetiva	Respecto a la técnica	10
Esfuerzo	Cuanto esfuerzo toma aplicarla (esfuerzo en aprendizaje, instalación, configuración y ejecución)?	15
Total		**100**

III. LISTADO DE TECNICAS DE TESTING

- Equivalence Class Partitioning
- Boundary Value Analysis
- Decision Tables
- State Transition Diagrams
- Orthogonal Arrays
- All Pairs Technique
- Fuzz testing
- Cause-effect graphing
- Desk checking
- Code walkthrough
- Formal Inspections
- Statement coverage
- Branch coverage
- Decision/Condition Coverage
- Function Coverage
- Flow Graph Notation
- Cyclomatic Complexity
- Deriving Test Cases
- Graph Matrices
- Simple Loops
- Nested loops
- Concatenated loops
- Unstructured loop

IV. RECOMENDADACION DE TECNICAS

En base a:
- Sección I: descripción y caracterización del proyecto
- Sección II: preferencias del equipo del proyecto respecto al desempeño de las técnicas a recomendar
- Sección III: Listado de técnicas de testing

Sírvase recomendar 3 técnicas de testing. La recomendación debe darse en forma de ranking, donde la técnica que se encuentre en la primera posición es aquella que usted recomienda con mayor énfasis y las de posiciones posteriores son aquellas que usted recomienda con menor énfasis. **MUY IMPORTANTE:** Ud. Debe sugerir exactamente 3 técnicas de testing. No debe añadir filas adicionales o dejar en blanco alguna de las 3 filas.

Posición	Técnica recomendada	¿Por qué la recomienda? (Relacione su recomendación con Sección I y II)
1.		
2.		
3.		

Apéndice 2 – MATRICES DE ANÁLISIS DE RECOMENDACIONES HECHAS POR SOTESTER

SIGESCOM (0)	Sim	Normalize	Equivale	Boundary	Decision T	State Tran	Orthogonal	All Pairs T	Fuzz testi	Cause-eff	Desk chec	Code walk	Formal Ins	Statemen	Branch co	Decision/Con	Function Co
Mejoras al Proceso de Bloqueos	0.667435	1									0.61	0.824			0.897		
TD CRM - Generación de Interfaces para Fuente $ST	0.632119	0.947086												0.887	0.9169	0.9914	0.922
Sistema de calculo de comisiones de agencias	0.626234	0.938269		0.498													
Gestion de Provisiones de Seguros	0.601096	0.900606		0.538	0.459												
PORTABILIDAD NUMERICA	0.596249	0.893343		0.551	0.2715			0.433					0.361				
Implementación de BanTotal en Mi Banco	0.570722	0.855097		0.641	0.85	0.83		0.211	0.652					0.9			
Sistema NMIC	0.568524	0.851805			0.8045	0.5645				0.738							
Sistema de Gestión de desempeño	0.558818	0.837262			0.676					0.5505			0.2825	0.3555			0.79
Arial	0.558323	0.83652	0.62	0.72	0.83											0.88	
Módulo de compra y venta de CDS	0.541824	0.8118											0.274				0.721
ADS Regulatorio	0.537473	0.805281	0.9055	0.9205													
SIGESTI	0.534447	0.800748	0.863		0.805												
Web de siniestros online	0.521243	0.780964			0.7005					0.6725							
Envío de Mensajes Telefonía	0.514113	0.770282	0.444		0.4175												
Modificaciòn de Nòmina de trabajador	0.504662	0.756121	0.704	0.756													
PROY: RO-ROPE	0.474874	0.711491			0.6595												
GUIDEWIRE - MÓDULO BILLING CENTER	0.467598	0.700589			0.539			0.456		0.4935							
Integración de Data Warehouse en Pacífico Seguros	0.456404	0.683817			0.781	0.725				0.686							
ATM - Banca - Comisiones Por Transacción	0.454754	0.681345			0.531			0.424									
Diferimiento de primas	0.424384	0.635843			0.69					0.7							
Separación OnLine de Diferidos	0.423712	0.634836			0.772					0.689			0.827				
PROYECTO CREO R1 - Módulo Pólicy	0.389696	0.583872		0.4995	0.5275												
Cobros por cargos automáticos	0.38762	0.58076	0.953	0.9415	0.909	0.89	0.915										
Bapi Fino	0.384033	0.575387			0.585												
Flujo de impresión tarjeta de crédito	0.383635	0.57479															
MQ - BROKER BCP - DIRECTV (PAGOS)	0.35778	0.536052	0.512														
Ins Global			7	9	18	4	1	4	1	7	1	1	4	3	2	2	3
Rating Global			0.7145	0.673944	0.656	0.752375	0.915	0.381	0.652	0.647071	0.61	0.824	0.436125	0.714167	**0.90695**	**0.9357**	0.811
Ins Mas Sim kNN50%			5	7	9	2	0	2	1	3	1	1	3	3	2	2	3
Rating Mas Sim kNN50%			0.7073	0.660643	0.646	0.69725		0.322	0.652	0.653667	0.61	0.824	0.305833	0.714167	**0.90695**	**0.9357**	0.811
Rating weigted			0.52016	0.527305	0.482535	0.550804	0.5313955	0.293901	0.557523	0.473154	0.61	**0.824**	0.326617	0.635766	**0.882692**	**0.837539396**	0.70665284
Weighted I>=3			7	9	18	**4**	1	4	1	7	1	1	4	**3**	2	2	**3**
Weighted I>=2			7	9	18	4	1	4	1	7	1	1	4	3	**2**	**2**	**3**

GLOBALNET (1)	Sim		Equivale	Boundary	Decision T	State Tran	Orthogonal	All Pairs T	Fuzz testi	Cause-eff	Desk chec	Code wall	Formal In	Statemen	Branch co	Decision/Con	Function Co	Flow Grap	Deriving Tes	Graph Ma
Gestion de Provisiones de Seguros	0.683702	1		0.54	0.43															
Sistema de calculo de comisiones de agencias	0.67394	0.985721		0.51																
Módulo de compra y venta de CDS	0.666477	0.974805											0.29				0.795			
SIGESTI	0.652158	0.953862	0.88		0.72															0.78
ADS Regulatorio	0.650947	0.952092	0.87	0.885																
Mejoras al Proceso de Bloqueos	0.62021	0.907134									0.68	0.89			0.89					
Arial	0.602375	0.881048	0.66	0.64	0.76											0.76		0.8		
Sistema de Gestión de desempeño	0.597321	0.873656			0.7					0.585			0.385	0.5			0.85			
TD CRM - Generación de Interfaces para Fuente $ST	0.594554	0.86961												0.932	0.92	1	0.953			
Implementación de BanTotal en Mi Banco	0.589517	0.862243		0.6	0.9	0.86		0.27	0.69					0.85						
Web de siniestros online	0.566613	0.828742			0.71					0.635										
Separación OnLine de Diferidos	0.565861	0.827642			0.8					0.62			0.87							
ATM - Banca - Comisiones Por Transacción	0.559159	0.817839			0.52			0.4												
Diferimiento de primas	0.54997	0.8044			0.68					0.71										
Sistema NMIC	0.542787	0.793894			0.83	0.63				0.79										
Cobros por cargos automáticos	0.526003	0.769345	0.96	0.97	0.97	0.92	0.93													
PORTABILIDAD NUMERICA	0.525529	0.768652		0.54	0.235			0.45					0.365							
Modificaciòn de Nòmina de trabajador	0.480256	0.702434	0.71	0.76																
Envío de Mensajes Telefonía	0.458159	0.670115	0.45		0.43															
GUIDEWIRE - MÓDULO BILLING CENTER	0.456568	0.667788			0.57			0.49		0.445										
Integración de Data Warehouse en Pacífico Seguros	0.450224	0.658508			0.78	0.71				0.68										
PROYECTO CREO R1 - Módulo Pólicy	0.44497	0.650824		0.47	0.55															
PROY: RO-ROPE	0.444235	0.64975			0.725													0.765		
Flujo de impresión tarjeta de crédito	0.429408	0.628062																	0.76	
Bapi Fino	0.377306	0.551857			0.65															
MQ - BROKER BCP - DIRECTV (PAGOS)	0.319367	0.467115	0.53																	
Ins Global			7	9	18	**4**	1	4	1	7	1	1	4	**3**	2	2	**3**	2	1	1
Rating Global			0.72286	0.657222	0.664444	0.78	**0.93**	0.4025	0.69	0.637857	0.68	**0.89**	0.4775	0.760667	**0.905**	0.88	0.866	0.7825	0.76	0.78
Ins Mas Sim			**4**	7	12	**3**	1	3	1	5	1	1	4	3	2	2	**3**	1	0	1
Rating Mas Sim			0.8425	0.669286	0.687917	0.803333	**0.93**	0.373333	0.69	0.668	0.68	**0.89**	0.4775	0.760667	**0.905**	0.88	0.866	0.8		0.78
Weighted			0.57652	0.551957	0.519596	0.604255	0.71549092	0.308263	0.594948	0.499104	0.616851	**0.807349**	0.404914	0.66007	**0.803695**	0.769603043	**0.78210533**	0.600948	0.47732744	0.744012
Weighted I>=3			7	9	18	**4**	1	4	1	7	1	1	4	**3**	2	2	**3**	2	1	1
Weighted I>=2			7	9	18	4	1	4	1	7	1	1	4	3	**2**	**2**	**3**	2	1	1

SISAP LAB (2)	Sim		Equivale	Boundary	Decision T	State Tran	Orthogonal	All Pairs T	Fuzz testi	Cause-eff	Desk chec	Code wall	Formal In	Statemen	Branch co	Decision/Con	Function Co	Flow Grap	Deriving Tes	Graph Ma
Sistema de calculo de comisiones de agencias	0.641578	1		0.535																
SIGESTI	0.634859	0.989528	0.86		0.81															0.85
Arial	0.626276	0.976151	0.52	0.43	0.82											0.92		0.5		
Gestion de Provisiones de Seguros	0.615782	0.959793		0.55	0.47															
Módulo de compra y venta de CDS	0.611046	0.952412											0.22				0.76			
ADS Regulatorio	0.599735	0.934782	0.895	0.91																
TD CRM - Generación de Interfaces para Fuente $ST	0.571566	0.890876												0.894	0.945	0.98	0.92			
Implementación de BanTotal en Mi Banco	0.566086	0.882334		0.61	0.91	0.88		0.35	0.61					0.85						
PORTABILIDAD NUMERICA	0.552779	0.861594		0.5	0.275			0.41					0.315							
Sistema de Gestión de desempeño	0.55023	0.857621			0.74					0.485			0.3	0.345			0.81			
Mejoras al Proceso de Bloqueos	0.547628	0.853565									0.54	0.78			0.9					
ATM - Banca - Comisiones Por Transacción	0.54523	0.849826			0.44			0.36												
Separación OnLine de Diferidos	0.545072	0.84958			0.74					0.69			0.81							
Sistema NMIC	0.514177	0.801427			0.875	0.61				0.84										
Web de siniestros online	0.512472	0.798768			0.715					0.71										
Diferimiento de primas	0.510885	0.796294			0.71					0.69										
Modificaciòn de Nòmina de trabajador	0.482309	0.751754	0.6	0.69																
Envío de Mensajes Telefonía	0.460387	0.717586	0.42		0.43															
GUIDEWIRE - MÓDULO BILLING CENTER	0.458543	0.714712			0.59			0.33		0.435										
Integración de Data Warehouse en Pacífico Seguros	0.450582	0.702304			0.83	0.71				0.54										
Flujo de impresión tarjeta de crédito	0.449727	0.700971																	0.92	
PROYECTO CREO R1 - Módulo Pólicy	0.446582	0.696068		0.385	0.545															
Cobros por cargos automáticos	0.44573	0.694741	0.9	0.965	0.91	0.91	0.92													
PROY: RO-ROPE	0.404053	0.62978			0.655													0.745		
Bapi Fino	0.351242	0.547466			0.58															
MQ - BROKER BCP - DIRECTV (PAGOS)	0.350617	0.546492	0.43																	
Ins Global			7	9	18	**4**	1	4	1	7	1	1	4	**3**	2	2	**3**	2	1	1
Rating Global			0.66071	0.619444	0.669167	0.7775	**0.92**	0.3625	0.61	0.627143	0.54	0.78	0.41125	0.696333	**0.9225**	**0.95**	0.83	0.6225	**0.92**	0.85
Ins Mas Sim			**3**	6	11	2	0	3	1	5	1	1	4	**3**	2	2	**3**	1	0	1
Rating Mas Sim			0.75833	0.589167	0.682273	0.745		0.373333	0.61	0.683	0.54	0.78	0.41125	0.696333	**0.9225**	**0.95**	0.83	0.5		**0.85**
Weighted			0.54399	0.528825	0.534593	0.599043	0.63916147	0.300966	0.538224	0.497438	0.460925	0.66578	0.356595	0.614102	**0.805043**	**0.885558486**	0.74603723	0.478631	0.64489353	**0.841099**
Weighted I>=3			7	9	18	**4**	1	4	1	7	1	1	4	**3**	2	2	**3**	2	1	1
Weighted I>=2			7	9	18	4	1	4	1	7	1	1	4	3	**2**	**2**	**3**	2	1	1

SISTEMA VIRTUAL DE AUTOEVALUACION (3)	Sim		Equivale	Boundary	Decision T	State Tran	Orthogonal	All Pairs T	Fuzz testi	Cause-eff	Desk chec	Code wall	Formal In	Statemen	Branch co	Decision/Con	Function Co	Flow Grap	Deriving Tes	Graph Ma
Mejoras al Proceso de Bloqueos	0.503603	1									0.65	0.88			0.9					
Sistema de calculo de comisiones de agencias	0.49531	0.983532		0.4875																
Sistema de Gestión de desempeño	0.4863	0.965641			0.67					0.595			0.3225	0.44			0.82			
Módulo de compra y venta de CDS	0.461128	0.915657											0.3				0.76			
Arial	0.460745	0.914897	0.69	0.735	0.84											0.84		0.775		
Sistema NMIC	0.449949	0.893459			0.81	0.6025				0.755										
Integración de Data Warehouse en Pacífico Seguros	0.447174	0.887949			0.77	0.73				0.66										
SIGESTI	0.442435	0.878539	0.865		0.77															0.815
Diferimiento de primas	0.432592	0.858995			0.695					0.72										
Cobros por cargos automáticos	0.43032	0.854482	0.965	0.95	0.94	0.905	0.925													
Separación OnLine de Diferidos	0.408277	0.810712			0.8					0.66			0.85							
PORTABILIDAD NUMERICA	0.402746	0.79973		0.565	0.245			0.44					0.3625							
Gestion de Provisiones de Seguros	0.398633	0.791562		0.555	0.46															
ADS Regulatorio	0.395446	0.785232	0.9	0.9125																
PROY: RO-ROPE	0.386231	0.766934			0.705													0.7125		
Implementación de BanTotal en Mi Banco	0.383433	0.761379		0.64	0.845	0.81		0.23	0.665					0.875						
Modificaciòn de Nòmina de trabajador	0.369479	0.733671	0.71	0.785																
TD CRM - Generación de Interfaces para Fuente $ST	0.367285	0.729313												0.896	0.923	0.996	0.94			
Envío de Mensajes Telefonía	0.363729	0.722254	0.455		0.4275															
Web de siniestros online	0.353003	0.700955			0.715					0.6625										
Bapi Fino	0.349367	0.693735			0.615															
Flujo de impresión tarjeta de crédito	0.347415	0.689859																	0.84	
GUIDEWIRE - MÓDULO BILLING CENTER	0.345554	0.686164			0.555			0.485		0.475										
MQ - BROKER BCP - DIRECTV (PAGOS)	0.342717	0.68053	0.515																	
PROYECTO CREO R1 - Módulo Pólicy	0.331687	0.658628		0.485	0.53															
ATM - Banca - Comisiones Por Transacción	0.284631	0.565188			0.525			0.4												
Ins Global			7	9	18	**4**	1	4	1	7	1	1	4	**3**	2	2	**3**	2	1	1
Rating Global			0.72857	0.679444	0.662083	0.761875	**0.925**	0.38875	0.665	0.646786	0.65	0.88	0.45875	0.737	**0.9115**	**0.918**	0.84	0.74375	0.84	0.815
Ins Mas Sim			0	0	0	0	0	0	0	0	1	1	0	0	1	0	0	0	0	0
Rating Mas Sim											**0.65**	**0.88**			**0.9**					
Weighted			0.58893	0.550446	0.530861	0.644134	**0.79039607**	0.271466	0.506317	0.53986	0.65	**0.88**	0.391281	0.581518	**0.786578**	0.747454773	0.72442657	0.627743	0.57948117	0.716009
Weighted I>=3			**7**	9	18	**4**	1	4	1	7	1	1	4	3	2	2	**3**	2	1	1
Weighted I>=2			7	9	18	4	1	4	1	7	1	1	4	3	**2**	**2**	**3**	2	1	1

Gestion de almacen (4)	Sim		Equivale	Boundary	Decision T	State Tran	Orthogonal	All Pairs T	Fuzz testi	Cause-eff	Desk chec	Code wall	Formal In	Statemen	Branch co	Decision/Con	Function Co	Flow Grap	Deriving Tes	Graph Ma
Mejoras al Proceso de Bloqueos	0.615693	1									0.605	0.829			0.902					
Sistema de calculo de comisiones de agencias	0.591192	0.960205		0.4985																
Sistema de Gestión de desempeño	0.588713	0.956179			0.701					0.596			0.263	0.347			0.793			
Implementación de BanTotal en Mi Banco	0.561086	0.911308		0.638	0.815	0.803		0.248	0.65					0.9						
Sistema NMIC	0.542747	0.881522			0.822	0.585				0.7629										
Arial	0.529106	0.859367	0.655	0.67	0.84											0.92		0.565		
Integración de Data Warehouse en Pacífico Seguros	0.515209	0.836796			0.78	0.712				0.653										
SIGESTI	0.512954	0.833132	0.877		0.82															0.85
Módulo de compra y venta de CDS	0.498069	0.808956											0.255				0.72			
TD CRM - Generación de Interfaces para Fuente $ST	0.495297	0.804455												0.8584	0.924	0.988	0.925			
Diferimiento de primas	0.485389	0.788363			0.698					0.706										
Gestion de Provisiones de Seguros	0.474902	0.771329		0.54	0.456															
Separación OnLine de Diferidos	0.471569	0.765916			0.783					0.709			0.817							
Cobros por cargos automáticos	0.467297	0.758977	0.948	0.947	0.906	0.899	0.92													
Modificaciòn de Nòmina de trabajador	0.466708	0.758021	0.69	0.757																
ADS Regulatorio	0.463971	0.753576	0.909	0.922																
Web de siniestros online	0.436939	0.709671			0.724					0.683										
ATM - Banca - Comisiones Por Transacción	0.434534	0.705764			0.524			0.408												
PROY: RO-ROPE	0.426338	0.692452			0.649													0.7		
Envío de Mensajes Telefonía	0.417705	0.678431	0.4555		0.413															
PORTABILIDAD NUMERICA	0.395002	0.641556		0.535	0.266			0.425					0.346							
PROYECTO CREO R1 - Módulo Pólicy	0.383041	0.622129		0.492	0.54															
GUIDEWIRE - MÓDULO BILLING CENTER	0.345254	0.560757			0.558			0.4235		0.4885										
Bapi Fino	0.338728	0.550157			0.578															
Flujo de impresión tarjeta de crédito	0.321893	0.522814																	0.92	
MQ - BROKER BCP - DIRECTV (PAGOS)	0.283619	0.46065	0.495																	
Ins Global			**7**	9	18	**4**	1	4	1	7	1	1	4	3	2	2	**3**	2	1	1
Rating Global			0.7185	0.666611	0.659611	0.74975	**0.92**	0.376125	0.65	0.656914	0.605	0.829	0.42025	0.7018	0.913	**0.954**	0.81266667	0.6325	**0.92**	0.85
Ins Mas Sim			2	3	**6**	**3**	0	1	1	**3**	1	1	1	2	1	1	1	1	0	1
Rating Mas Sim			0.766	0.602167	0.796333	0.7		0.248	0.65	0.670633	0.605	0.829	0.263	0.6235	**0.902**	**0.92**	0.793	0.565		**0.85**
Weighted			0.53688	0.521007	0.507523	0.631397	0.69825914	0.256025	0.59235	0.521011	0.605	**0.829**	0.326373	0.614172	**0.822658**	**0.792709508**	0.6949398	0.485129	0.48098857	0.708162
Weighted I>=3			7	9	18	**4**	1	4	1	7	1	1	4	**3**	2	2	**3**	2	1	1
Weighted I>=2			7	9	18	4	1	4	1	7	1	1	4	3	**2**	**2**	**3**	2	1	1

Gestion produccion operarios (5)	Sim		Equivale	Boundary	Decision T	State Tran	Orthogonal A	All Pairs T	Fuzz testi	Cause-eff	Desk chec	Code wall	Formal In	Statemen	Branch co	Decision/Con	Function Co	Flow Grap	Deriving Tes	Graph Ma
Módulo de compra y venta de CDS	0.607865	1											0.27				0.701			
Sistema de calculo de comisiones de agencias	0.588562	0.968245		0.512																
ATM - Banca - Comisiones Por Transacción	0.587371	0.966286			0.51			0.457												
Implementación de BanTotal en Mi Banco	0.582928	0.958977		0.643	0.837	0.832		0.2	0.653					0.925						
Separación OnLine de Diferidos	0.569837	0.937441			0.756					0.682			0.805							
Arial	0.560187	0.921566	0.62	0.65	0.78											0.88		0.555		
Mejoras al Proceso de Bloqueos	0.558	0.917967									0.603	0.821			0.907					
Envío de Mensajes Telefonía	0.541512	0.890843	0.449		0.4265															
Cobros por cargos automáticos	0.524328	0.862573	0.957	0.9275	0.883	0.891	0.915													
ADS Regulatorio	0.514842	0.846968	0.9065	0.92																
SIGESTI	0.51313	0.844153	0.842		0.81															0.84
Sistema de Gestión de desempeño	0.508865	0.837135			0.684					0.5285			0.2645	0.3375			0.781			
Flujo de impresión tarjeta de crédito	0.507023	0.834105																	0.88	
Gestion de Provisiones de Seguros	0.500699	0.823701		0.534	0.471															
Modificaciòn de Nòmina de trabajador	0.490041	0.806167	0.686	0.75																
Diferimiento de primas	0.477241	0.785111			0.6985					0.703										
Integración de Data Warehouse en Pacífico Seguros	0.464057	0.763422			0.806	0.756				0.685										
Sistema NMIC	0.462743	0.761261			0.792	0.529				0.7148										
TD CRM - Generación de Interfaces para Fuente $ST	0.433206	0.712669												0.8682	0.9275	0.9871	0.921			
GUIDEWIRE - MÓDULO BILLING CENTER	0.417532	0.686883			0.525			0.453		0.5155										
PROYECTO CREO R1 - Módulo Pólicy	0.382815	0.62977		0.5015	0.539															
PORTABILIDAD NUMERICA	0.371882	0.611784		0.542	0.282			0.44					0.371							
MQ - BROKER BCP - DIRECTV (PAGOS)	0.350546	0.576684	0.498																	
Bapi Fino	0.350014	0.575809			0.577															
Web de siniestros online	0.345576	0.568508			0.704					0.68										
PROY: RO-ROPE	0.336356	0.55334			0.6265													0.6605		
Ins Global			7	9	18	**4**	1	4	1	7	1	1	4	**3**	2	2	**3**	2	1	1
Rating Global			0.70836	0.664444	0.650417	0.752	**0.915**	0.3875	0.653	0.644114	0.603	0.821	0.427625	0.710233	**0.91725**	**0.93355**	0.801	0.60775	0.88	0.84
Ins Mas Sim			**5**	**6**	**9**	2	1	2	1	2	1	1	3	2	1	1	2	1	1	1
Rating Mas Sim			0.7549	0.69775	0.684167	0.8615	**0.915**	0.3285	0.653	0.60525	0.603	0.821	0.4465	0.63125	**0.907**	**0.88**	0.741	0.555	**0.88**	0.84
Weighted			0.58794	0.553614	0.512627	0.636569	**0.7892547**	0.303433	0.626212	0.491637	0.553534	**0.753651**	0.368259	0.596109	0.746799	**0.757226931**	0.67039019	0.438475	0.73401251	0.709088
Weighted I>=3			7	9	18	**4**	1	4	1	7	1	1	4	**3**	2	2	**3**	2	1	1
Weighted I>=2			7	9	18	4	1	4	1	7	1	1	4	3	**2**	**2**	**3**	2	1	1

INTEGRACION JUEGOS DE AZAR (6)	Sim		Equivale	Boundary	Decision T	State Tran	Orthogonal A	All Pairs T	Fuzz testi	Cause-eff	Desk chec	Code wall	Formal In	Statemen	Branch co	Decision/Con	Function Co	Flow Grap	Deriving Tes	Graph Ma
Sistema de calculo de comisiones de agencias	0.620091	1		0.5																
Módulo de compra y venta de CDS	0.576722	0.930061											0.22				0.74			
Arial	0.555561	0.895935	0.34	0.56	0.84											0.84		0.6		
Gestion de Provisiones de Seguros	0.547769	0.883369		0.55	0.48															
SIGESTI	0.546845	0.881879	0.85		0.79															0.84
Sistema de Gestión de desempeño	0.544563	0.878198			0.7					0.41			0.35	0.325			0.78			
ADS Regulatorio	0.528003	0.851493	0.895	0.91																
Sistema NMIC	0.527286	0.850337			0.8	0.57				0.76										
Separación OnLine de Diferidos	0.525724	0.847818			0.7					0.69			0.82							
Implementación de BanTotal en Mi Banco	0.520097	0.838743		0.63	0.95	0.9		0.27	0.61					0.85						
Diferimiento de primas	0.516302	0.832623			0.67					0.64										
Cobros por cargos automáticos	0.508563	0.820143	0.89	0.965	0.92	0.88	0.9													
Envío de Mensajes Telefonía	0.502952	0.811095	0.38		0.42															
Modificaciòn de Nòmina de trabajador	0.499843	0.80608	0.63	0.65																
PORTABILIDAD NUMERICA	0.493859	0.796431		0.52	0.29			0.42					0.32							
Mejoras al Proceso de Bloqueos	0.487621	0.786371									0.51	0.71			0.88					
Flujo de impresión tarjeta de crédito	0.483128	0.779125																	0.84	
ATM - Banca - Comisiones Por Transacción	0.48127	0.776128			0.46			0.37												
TD CRM - Generación de Interfaces para Fuente $ST	0.472021	0.761213												0.91	0.91	0.985	0.899			
Web de siniestros online	0.440249	0.709976			0.635					0.68										
PROYECTO CREO R1 - Módulo Pólicy	0.433348	0.698847		0.395	0.48															
Integración de Data Warehouse en Pacífico Seguros	0.412662	0.665486			0.83	0.72				0.63										
GUIDEWIRE - MÓDULO BILLING CENTER	0.380614	0.613803			0.56			0.37		0.43										
Bapi Fino	0.380302	0.613301			0.57															
MQ - BROKER BCP - DIRECTV (PAGOS)	0.36367	0.586479	0.44																	
PROY: RO-ROPE	0.363056	0.585489			0.68													0.765		
Ins Global			7	9	18	**4**	1	4	1	7	1	1	4	**3**	2	2	**3**	2	1	1
Rating Global			0.63214	0.631111	0.654167	0.7675	**0.9**	0.3575	0.61	0.605714	0.51	0.71	0.4275	0.695	**0.895**	**0.9125**	0.80633333	0.6825	0.84	0.84
Ins Mas Sim			5	**6**	**10**	**3**	1	1	1	4	0	0	3	2	0	1	2	1	0	1
Rating Mas Sim			0.671	0.685833	0.727	0.783333	**0.9**	0.27	0.61	0.625			0.463333	0.5875		**0.84**	0.76	0.6		**0.84**
Weighted			0.51719	0.532936	0.51276	0.610109	**0.73812915**	0.268809	0.511633	0.470024	0.401049	0.558324	0.365513	0.563683	0.692355	**0.751190009**	0.68585677	0.49273	0.65446489	**0.740778**
Weighted I>=3			7	9	18	**4**	1	4	1	7	1	1	4	**3**	2	2	**3**	2	1	1
Weighted I>=2			7	9	18	4	1	4	1	7	1	1	4	3	**2**	**2**	**3**	2	1	1

control de tareo (7)	Sim		Equivale	Boundary	Decision T	State Tran	Orthogonal	All Pairs T	Fuzz testi	Cause-eff	Desk chec	Code wall	Formal In	Statemen	Branch co	Decision/Con	Function Co	Flow Grap	Deriving Tes	Graph Ma
Mejoras al Proceso de Bloqueos	0.668373	1									0.52	0.72			0.905					
TD CRM - Generación de Interfaces para Fuente $ST	0.632245	0.945946												0.894	0.938	0.978	0.901			
SIGESTI	0.574277	0.859216	0.77		0.76															0.81
Módulo de compra y venta de CDS	0.571412	0.85493											0.24				0.71			
Implementación de BanTotal en Mi Banco	0.564271	0.844246		0.63	0.95	0.92		0.22	0.63					0.9						
Sistema de calculo de comisiones de agencias	0.560274	0.838265		0.525																
Sistema de Gestión de desempeño	0.558962	0.836301			0.69					0.315			0.36	0.345			0.77			
Gestion de Provisiones de Seguros	0.534486	0.799682		0.55	0.52															
Web de siniestros online	0.530203	0.793274			0.62					0.68										
Envío de Mensajes Telefonía	0.526036	0.787039	0.375		0.46															
Sistema NMIC	0.523054	0.782578			0.745	0.475				0.67										
Modificaciòn de Nòmina de trabajador	0.51803	0.775061	0.58	0.64																
ADS Regulatorio	0.516774	0.773181	0.885	0.895																
Diferimiento de primas	0.503724	0.753656			0.68					0.645										
Arial	0.503337	0.753077	0.31	0.39	0.66											0.76		0.6		
ATM - Banca - Comisiones Por Transacción	0.500758	0.749219			0.39			0.47												
PORTABILIDAD NUMERICA	0.495393	0.741193		0.52	0.305			0.45					0.36							
GUIDEWIRE - MÓDULO BILLING CENTER	0.488299	0.730578			0.51			0.42		0.475										
Separación OnLine de Diferidos	0.482659	0.72214			0.65					0.62			0.78							
Cobros por cargos automáticos	0.473169	0.707942	0.91	0.925	0.87	0.88	0.9													
Integración de Data Warehouse en Pacífico Seguros	0.472211	0.706508			0.9	0.83				0.65										
PROYECTO CREO R1 - Módulo Pólicy	0.453425	0.6784		0.385	0.495															
Bapi Fino	0.393118	0.588171			0.57															
PROY: RO-ROPE	0.349468	0.522864			0.625													0.66		
Flujo de impresión tarjeta de crédito	0.348951	0.522089																	0.76	
MQ - BROKER BCP - DIRECTV (PAGOS)	0.324968	0.486207	0.41																	
Ins Global			7	9	18	**4**	1	4	1	7	1	1	4	**3**	2	2	**3**	2	1	1
Rating Global			0.60571	0.606667	0.633333	0.77625	**0.9**	0.39	0.63	0.579286	0.52	0.72	0.435	0.713	**0.9215**	**0.869**	0.79366667	0.63	0.76	0.81
Ins Mas Sim			5	6	**10**	2	0	2	1	4	1	1	2	**3**	2	2	**3**	1	0	1
Rating Mas Sim			0.584	0.605	0.6475	0.6975		0.345	0.63	0.5775	0.52	0.72	0.3	0.713	**0.9215**	**0.869**	0.79366667	0.6		**0.81**
Weighted			0.45251	0.466108	0.472455	0.589455	0.63714739	0.294562	0.531875	0.438183	0.52	**0.72**	0.334088	0.63134	**0.896149**	**0.748736944**	0.70108318	0.398468	0.39678796	0.695965
Weighted I>=3			7	9	18	**4**	1	4	1	7	1	1	4	**3**	2	2	**3**	2	1	1
Weighted I>=2			7	9	18	4	1	4	1	7	1	1	4	3	**2**	**2**	**3**	2	1	1

Sotester (8)	Sim		Equivale	Boundary	Decision T	State Tran	Orthogonal A	All Pairs T	Fuzz testi	Cause-eff	Desk chec	Code wall	Formal In	Statemen	Branch co	Decision/Con	Function Co	Flow Grap	Deriving Tes	Graph Ma
Mejoras al Proceso de Bloqueos	0.546016	1									0.595	0.815			0.8975					
TD CRM - Generación de Interfaces para Fuente $ST	0.528036	0.967069												0.91	0.9255	0.9895	0.9195			
Sistema de calculo de comisiones de agencias	0.52404	0.959751		0.495																
PORTABILIDAD NUMERICA	0.494857	0.906305		0.56	0.2675			0.425					0.3525							
Sistema de Gestión de desempeño	0.48953	0.896549			0.65					0.4975			0.33	0.37			0.7975			
ADS Regulatorio	0.478098	0.87561	0.9025	0.92																
Módulo de compra y venta de CDS	0.477666	0.874821											0.275				0.735			
SIGESTI	0.474155	0.86839	0.845		0.8															0.845
Web de siniestros online	0.473666	0.867494			0.695					0.695										
Integración de Data Warehouse en Pacífico Seguros	0.460566	0.843503			0.79	0.73				0.655										
Separación OnLine de Diferidos	0.459586	0.841708			0.75					0.675			0.83							
Diferimiento de primas	0.450938	0.825869			0.6925					0.6975										
Gestion de Provisiones de Seguros	0.44909	0.822484		0.545	0.47															
Sistema NMIC	0.44715	0.818932			0.82	0.5775				0.757										
Modificaciòn de Nòmina de trabajador	0.434715	0.796158	0.66	0.74																
Bapi Fino	0.430563	0.788554			0.58															
Arial	0.42842	0.784629	0.58	0.695	0.83											0.88		0.675		
Cobros por cargos automáticos	0.421965	0.772806	0.935	0.945	0.915	0.885	0.91													
Implementación de BanTotal en Mi Banco	0.418111	0.765748		0.635	0.885	0.845		0.23	0.645					0.85						
GUIDEWIRE - MÓDULO BILLING CENTER	0.413354	0.757035			0.5425			0.4525		0.46										
PROY: RO-ROPE	0.399359	0.731405			0.6675													0.7125		
Envío de Mensajes Telefonía	0.394736	0.722938	0.415		0.435															
ATM - Banca - Comisiones Por Transacción	0.383909	0.703108			0.495			0.425												
Flujo de impresión tarjeta de crédito	0.367196	0.6725																	0.88	
MQ - BROKER BCP - DIRECTV (PAGOS)	0.361639	0.662323	0.495																	
PROYECTO CREO R1 - Módulo Pólicy	0.269253	0.493122		0.4525	0.51															
Ins Global			7	9	18	**4**	1	4	1	7	1	1	4	**3**	2	2	**3**	2	1	1
Rating Global			0.69036	0.665278	0.655278	0.759375	**0.91**	0.383125	0.645	0.633857	0.595	0.815	0.446875	0.71	**0.9115**	**0.93475**	0.81733333	0.69375	0.88	0.845
Ins Mas Sim			0	1	0	0	0	0	0	0	1	1	0	1	2	1	1	0	0	0
Rating Mas Sim				0.495							0.595	0.815		0.91	**0.9115**	**0.9895**	**0.9195**			
Weighted			0.55072	0.53451	0.519822	0.60492	0.70325338	0.30067	0.493908	0.530543	0.595	**0.815**	0.388632	0.620881	**0.896261**	**0.823694399**	0.74907041	0.525376	0.59179967	0.73379
Weighted I>=3			7	9	18	**4**	1	4	1	7	1	1	4	**3**	2	2	**3**	2	1	1
Weighted I>=2			7	9	18	4	1	4	1	7	1	1	4	3	**2**	**2**	**3**	2	1	1

detracciones (9)	Sim		Equivale	Boundary	Decision T	State Tran	Orthogonal	All Pairs T	Fuzz testi	Cause-eff	Desk chec	Code wall	Formal In	Statemen	Branch co	Decision/Con	Function Co	Flow Grap	Deriving Tes	Graph Ma
Mejoras al Proceso de Bloqueos	0.624823	1									0.48	0.74			0.905					
TD CRM - Generación de Interfaces para Fuente $ST	0.603758	0.966286												0.884	0.943	0.978	0.909			
Implementación de BanTotal en Mi Banco	0.578066	0.925167		0.66	0.95	0.92		0.27	0.57					0.9						
Módulo de compra y venta de CDS	0.577977	0.925025											0.24				0.74			
Sistema de calculo de comisiones de agencias	0.569487	0.911438		0.545																
Modificaciòn de Nòmina de trabajador	0.55788	0.892861	0.6	0.66																
Sistema de Gestión de desempeño	0.549301	0.879131			0.73					0.36			0.295	0.32			0.78			
SIGESTI	0.54588	0.873655	0.8		0.79															0.83
Envío de Mensajes Telefonía	0.54259	0.86839	0.415		0.445															
PORTABILIDAD NUMERICA	0.53864	0.862068		0.51	0.31			0.44					0.34							
Sistema NMIC	0.533469	0.853792			0.79	0.51				0.75										
Arial	0.530501	0.849041	0.39	0.41	0.84											0.84		0.55		
Gestion de Provisiones de Seguros	0.523548	0.837915		0.57	0.53															
GUIDEWIRE - MÓDULO BILLING CENTER	0.522579	0.836363			0.55			0.33		0.48										
ADS Regulatorio	0.500366	0.800813	0.91	0.92																
ATM - Banca - Comisiones Por Transacción	0.495764	0.793447			0.4			0.37												
PROYECTO CREO R1 - Módulo Pólicy	0.480412	0.768876		0.37	0.51															
Web de siniestros online	0.480018	0.768246			0.655					0.705										
Cobros por cargos automáticos	0.458489	0.733791	0.9	0.94	0.88	0.9	0.91													
Separación OnLine de Diferidos	0.455511	0.729024			0.69					0.67			0.79							
Diferimiento de primas	0.451984	0.72338			0.72					0.675										
Bapi Fino	0.433662	0.694056			0.58															
Integración de Data Warehouse en Pacífico Seguros	0.428427	0.685677			0.89	0.8				0.53										
PROY: RO-ROPE	0.394531	0.631428			0.66													0.69		
Flujo de impresión tarjeta de crédito	0.376772	0.603007																	0.84	
MQ - BROKER BCP - DIRECTV (PAGOS)	0.340729	0.545322	0.39																	
Ins Global			7	9	18	**4**	1	4	1	7	1	1	4	**3**	2	2	**3**	2	1	1
Rating Global			0.62929	0.620556	0.662222	0.7825	**0.91**	0.3525	0.57	0.595714	0.48	0.74	0.41625	0.701333	**0.924**	**0.909**	0.80966667	0.62	0.84	0.83
Ins Mas Sim			5	7	**9**	2	0	3	1	3	1	1	3	**3**	2	2	**3**	1	0	1
Rating Mas Sim			0.623	0.610714	0.659444	0.715		0.346667	0.57	0.53	0.48	0.74	0.291667	0.701333	**0.924**	**0.909**	0.80966667	0.55		**0.83**
Weighted			0.504	0.519222	0.52601	0.623885	0.66774965	0.29967	0.527345	0.462862	0.48	0.74	0.337595	0.656056	**0.908104**	**0.829111113**	**0.74953163**	0.451329	0.50652559	0.725133
Weighted I>=3			7	9	18	**4**	1	4	1	7	1	1	4	**3**	2	2	**3**	2	1	1
Weighted I>=2			7	9	18	4	1	4	1	7	1	1	4	3	**2**	**2**	**3**	2	1	1

Control de materiales en producción (10)	Sim		Equivale	Boundary	Decision T	State Tran	Orthogonal A	All Pairs T	Fuzz testi	Cause-eff	Desk chec	Code wall	Formal In	Statemen	Branch co	Decision/Con	Function Co	Flow Grap	Deriving Tes	Graph Ma
TD CRM - Generación de Interfaces para Fuente $ST	0.565984	1												0.8948	0.9346	0.9866	0.9202			
Módulo de compra y venta de CDS	0.558277	0.986382											0.277				0.72			
Mejoras al Proceso de Bloqueos	0.557772	0.985489									0.595	0.819			0.9065					
Sistema de calculo de comisiones de agencias	0.533315	0.942279		0.5215																
Envío de Mensajes Telefonía	0.52524	0.928012	0.44		0.4325															
Cobros por cargos automáticos	0.519246	0.917421	0.951	0.929	0.887	0.893	0.917													
Sistema NMIC	0.517874	0.914997			0.812	0.5385				0.7363										
Separación OnLine de Diferidos	0.49174	0.868823			0.752					0.667			0.811							
Implementación de BanTotal en Mi Banco	0.490538	0.866699		0.636	0.873	0.859		0.222	0.64					0.915						
Sistema de Gestión de desempeño	0.490257	0.866202			0.684					0.4935			0.2665	0.347			0.789			
ADS Regulatorio	0.48481	0.856579	0.903	0.9175																
PORTABILIDAD NUMERICA	0.477702	0.84402		0.545	0.282			0.435					0.3675							
Diferimiento de primas	0.461104	0.814693			0.7035					0.7055										
SIGESTI	0.453421	0.801119	0.833		0.806															0.838
Gestion de Provisiones de Seguros	0.449784	0.794693		0.541	0.476															
Arial	0.442123	0.781157	0.6	0.61	0.78											0.88		0.56		
Integración de Data Warehouse en Pacífico Seguros	0.43548	0.769421			0.812	0.758				0.652										
ATM - Banca - Comisiones Por Transacción	0.430098	0.759911			0.493			0.439												
Modificaciòn de Nòmina de trabajador	0.420443	0.742853	0.669	0.747																
Flujo de impresión tarjeta de crédito	0.416485	0.735859																	0.88	
MQ - BROKER BCP - DIRECTV (PAGOS)	0.377855	0.667607	0.489																	
PROYECTO CREO R1 - Módulo Pólicy	0.37238	0.657933		0.476	0.543															
Web de siniestros online	0.342329	0.604838			0.702					0.684										
Bapi Fino	0.337045	0.595503			0.583															
GUIDEWIRE - MÓDULO BILLING CENTER	0.328931	0.581166			0.53			0.4415		0.505										
PROY: RO-ROPE	0.209146	0.369526			0.6375													0.6695		
Ins Global			7	9	18	**4**	1	4	1	7	1	1	4	**3**	2	2	**3**	2	1	1
Rating Global			0.69786	0.658111	0.654917	0.762125	**0.917**	0.384375	0.64	0.634757	0.595	0.819	0.4305	0.718933	**0.92055**	**0.9333**	0.80973333	0.61475	0.88	0.838
Ins Mas Sim			2	2	**3**	2	1	0	0	1	1	1	1	1	2	1	2	0	0	0
Rating Mas Sim			0.6955	0.72525	0.7105	0.71575	**0.917**			0.7363	0.595	0.819	0.277	0.8948	**0.92055**	**0.9866**	0.8201			
Weighted			0.57339	0.546147	0.504279	0.659925	**0.84127522**	0.287435	0.554687	0.494902	0.586366	0.807116	0.379716	0.6628	**0.913973**	**0.837009099**	0.77127617	0.342423	0.64755572	0.671337
Weighted I>=3			7	9	18	**4**	1	4	1	7	1	1	4	**3**	2	2	**3**	2	1	1
Weighted I>=2			7	9	18	4	1	4	1	7	1	1	4	3	**2**	**2**	**3**	2	1	1

Apéndice 3 - PROYECTOS OBJETIVO

PROYECTO 0

Descripción y caracterización del proyecto
SIGESCOM: Sistema de gestión comercial para mypes: gestión de ventas, compras, clientes, proveedores, facturación, gastos y costos.

Atributo	Descripción	Valor
Estático o dinámico	Tipo de técnica de testing requerida	Dinámico
Tipo de Software	Tipo de software a probar	Web
Fase de ciclo de vida	Fase de desarrollo o ciclo de vida en el cual se realizarán las pruebas	Pruebas Unitarias Pruebas de componentes Pruebas del sistema
Escalabilidad	Tamaño del sistema a ser probado (LoC)	>= 3001
Entorno: lenguaje de programación	Lenguaje de programación usado para el desarrollo de software	C#
Entrada	¿Qué entradas están disponibles para el proceso de testing?	Código fuente Programa ejecutable
Conocimiento	¿Qué conocimientos tiene el equipo de pruebas?	Dominio de la aplicación, Conocimiento acerca de cómo identificar el estado del sistema a probar, Paradigma orientado a objetos, Programación, Trabajo en equipo, Liderazgo
Experiencia	¿Qué experiencia tiene el equipo de pruebas?	Web testing
Salida	Salidas que el equipo espera producto de la aplicación de las técnicas de pruebas	Casos de prueba Defectos
Tipos de defectos	Principales tipos de defectos que se quiere detectar	Verificación o Checking Algoritmos Sincronización/Serialización
Aplicabilidad en tareas	Actividades de pruebas que se quiere realizar	Pruebas unitarias Pruebas del sistema
Obtención de herramienta	Tipo de licenciamiento que prefiere para la herramienta de pruebas	Comercial

Preferencias del equipo de proyecto respecto al desempeño de las técnicas a recomendar

Atributo	Descripción	Peso
Completitud	Cobertura provista por los casos de prueba	15
Eficacia	Capacidad de encontrar defectos	20
Tamaño del banco de pruebas	Nivel de adecuación del número de casos de prueba generados por unidad de tamaño de software	12
Interacción	Cuán adecuados son los modos de interacción; (Por ejemplo: navegación, consultas, refinamiento sucesivo, etc.) soportados por la técnica	5
Guía o participación del usuario	Cuan apropiada es la guía o participación que la técnica requiere del usuario. Por ejemplo: Nada apropiada (la técnica es completamente manual), evaluación manual de las salidas, selección de entradas apropiadas, definición de patrones, filtrado, etc.	5
Fuentes de información	Disponibilidad de fuentes de información acerca de la técnica	8
Comprensibilidad	Si la técnica es o no es fácil de entender	10
Satisfacción subjetiva	Respecto a la técnica	10
Esfuerzo	¿Cuánto esfuerzo toma aplicarla (esfuerzo en aprendizaje, instalación, configuración y ejecución)?	15
Total		**100**

PROYECTO 1

Descripción y caracterización del proyecto
GLOBALNET: Sistema web comercial para tienda de artículos y servicios de telecomunicaciones.

Caracterización		
Atributo	**Descripción**	**Valor**
Estático o dinámico	Tipo de técnica de testing requerida	Dinámica
Tipo de Software	Tipo de software a probar	Web
Fase de ciclo de vida	Fase de desarrollo o ciclo de vida en el cual se realizaran las pruebas	Pruebas de aceptación
Escalabilidad	Tamaño del sistema a ser probado (LoC)	>=3001
Entorno: lenguaje de programación	Lenguaje de programación usado para el desarrollo de software	PHP
Entrada	¿Qué entradas están disponibles para el proceso de testing?	Código Fuente Programa ejecutable Escenarios de ejecución Requisitos de la aplicación
Conocimiento	¿Qué conocimientos tiene el equipo de pruebas?	Conocimiento acerca de cómo identificar el estado del sistema a probar Identificación de escenarios Paradigm orientado a objetos Codificación Trabajo en equipo
Experiencia	¿Qué experiencia tiene el equipo de pruebas?	Pruebas de requisites Web testing
Salida	Salidas que el equipo espera producto de la aplicación de las técnicas de pruebas	Defectos
Tipos de defectos	Principales tipos de defectos que se quiere detectar	Interfaz Función
Aplicabilidad en tareas	Actividades de pruebas que se quiere realizar	Pruebas de requisitos
Obtención de herramienta	Tipo de licenciamiento que prefiere para la herramienta de pruebas	Open source

Preferencias del equipo de proyecto respecto al desempeño de las técnicas a recomendar

Atributo	**Descripción**	**Peso**
Completitud	Cobertura provista por los casos de prueba	30
Eficacia	Capacidad de encontrar defectos	30
Tamaño del banco de pruebas	Nivel de adecuación del número de casos de prueba generados por unidad de tamaño de software	10
Interacción	Cuán adecuados son los modos de interacción; (Por ejemplo: navegación, consultas, refinamiento sucesivo, etc.) soportados por la técnica	0
Guía o participación del usuario	Cuan apropiada es la guía o participación que la técnica requiere del usuario. Por ejemplo: Nada apropiada (la técnica es completamente manual), evaluación manual de las salidas, selección de entradas apropiadas, definición de patrones, filtrado, etc.	0
Fuentes de información	Disponibilidad de fuentes de información acerca de la técnica	0
Comprensibilidad	Si la técnica es o no es fácil de entender	0
Satisfacción subjetiva	Respecto a la técnica	0
Esfuerzo	¿Cuánto esfuerzo toma aplicarla (esfuerzo en aprendizaje, instalación, configuración y ejecución)?	30
Total		**100**

PROYECTO 2

Descripción y caracterización del proyecto
SISAP / LAB: Sistema de información para entidades de salud pública / módulo de reportes clínicos - laboratorio

Caracterización		
Atributo	**Descripción**	**Valor**
Estático o dinámico	Tipo de técnica de testing requerida	Dinámica
Tipo de Software	Tipo de software a probar	Web
Fase de ciclo de vida	Fase de desarrollo o ciclo de vida en el cual se realizarán las pruebas	Pruebas de aceptación
Escalabilidad	Tamaño del sistema a ser probado (LoC)	501 – 1000
Entorno: lenguaje de programación	Lenguaje de programación usado para el desarrollo de software	Java
Entrada	¿Qué entradas están disponibles para el proceso de testing?	Código Fuente Programa ejecutable
Conocimiento	¿Qué conocimientos tiene el equipo de pruebas?	Paradigma orientado a objetos Codificación Trabajo en equipo Liderazgo
Experiencia	¿Qué experiencia tiene el equipo de pruebas?	Testing de requisitos, Web testing
Salida	Salidas que el equipo espera producto de la aplicación de las técnicas de pruebas	Defectos
Tipos de defectos	Principales tipos de defectos que se quiere detectar	Interfaz Función Documentación
Aplicabilidad en tareas	Actividades de pruebas que se quiere realizar	Pruebas de requisites
Obtención de herramienta	Tipo de licenciamiento que prefiere para la herramienta de pruebas	Open source

Preferencias del equipo de proyecto respecto al desempeño de las tecnicas a recomendar

Atributo	**Descripción**	**Peso**
Completitud	Cobertura provista por los casos de prueba	20
Eficacia	Capacidad de encontrar defectos	30
Tamaño del banco de pruebas	Nivel de adecuación del número de casos de prueba generados por unidad de tamaño de software	0
Interacción	Cuán adecuados son los modos de interacción; (Por ejemplo: navegación, consultas, refinamiento sucesivo, etc.) soportados por la técnica	10
Guía o participación del usuario	Cuan apropiada es la guía o participación que la técnica requiere del usuario. Por ejemplo: Nada apropiada (la técnica es completamente manual), evaluación manual de las salidas, selección de entradas apropiadas, definición de patrones, filtrado, etc.	30
Fuentes de información	Disponibilidad de fuentes de información acerca de la técnica	0
Comprensibilidad	Si la técnica es o no es fácil de entender	0
Satisfacción subjetiva	Respecto a la técnica	0
Esfuerzo	¿Cuánto esfuerzo toma aplicarla (esfuerzo en aprendizaje, instalación, configuración y ejecución)?	10
Total		**100**

PROYECTO 3

Descripción y caracterización del proyecto

SISTEMA VIRTUAL DE AUTOEVALUACION: Software web para gestionar el proceso de autoevaluación de las carreras profesionales de una universidad en base a modelos de calidad. Permite registrar y evaluar estándares, mediante documentación, encuestas, cuestionarios, indicadores de gestión, El software se integra con otros sistemas que proporcionan datos del personal docente, estudiantes y trabajadores administrativos, así como datos para el cálculo de indicadores de gestión.

Caracterización		
Atributo	**Descripción**	**Valor**
Estático o dinámico	Tipo de técnica de testing requerida	Estática, Dinámica
Tipo de Software	Tipo de software a probar	Web
Fase de ciclo de vida	Fase de desarrollo o ciclo de vida en el cual se realizarán las pruebas	Pruebas unitarias, Pruebas de componentes, Pruebas de integración de componentes, Pruebas de integración del sistema, Pruebas de sistema.
Escalabilidad	Tamaño del sistema a ser probado (LoC)	> 3001
Entorno: lenguaje de programación	Lenguaje de programación usado para el desarrollo de software	Java
Entrada	¿Qué entradas están disponibles para el proceso de testing?	Código Fuente, Programa ejecutable
Conocimiento	¿Qué conocimientos tiene el equipo de pruebas?	Dominio de aplicación, Conocimiento acerca de cómo identificar el estado del sistema a probar, Paradigma orientado a objetos, Codificación ,Trabajo en equipo
Experiencia	¿Qué experiencia tiene el equipo de pruebas?	--
Salida	Salidas que el equipo espera producto de la aplicación de las técnicas de pruebas	Casos de prueba, escenarios de pruebas, Defectos
Tipos de defectos	Principales tipos de defectos que se quiere detectar	Algoritmo, Función, Timming Serializacion
Aplicabilidad en tareas	Actividades de pruebas que se quiere realizar	Pruebas unitarias, Pruebas del sistema
Obtención de herramienta	Tipo de licenciamiento que prefiere para la herramienta de pruebas	Open source

Preferencias del equipo de proyecto respecto al desempeño de las técnicas a recomendar

Atributo	**Descripción**	**Peso**
Completitud	Cobertura provista por los casos de prueba	25
Eficacia	Capacidad de encontrar defectos	25
Tamaño del banco de pruebas	Nivel de adecuación del número de casos de prueba generados por unidad de tamaño de software	10
Interacción	Cuán adecuados son los modos de interacción; (Por ejemplo: navegación, consultas, refinamiento sucesivo, etc.) soportados por la técnica	0
Guía o participación del usuario	Cuan apropiada es la guía o participación que la técnica requiere del usuario. Por ejemplo: Nada apropiada (la técnica es completamente manual), evaluación manual de las salidas, selección de entradas apropiadas, definición de patrones, filtrado, etc.	0
Fuentes de información	Disponibilidad de fuentes de información acerca de la técnica	10
Comprensibilidad	Si la técnica es o no es fácil de entender	10
Satisfacción subjetiva	Respecto a la técnica	0
Esfuerzo	¿Cuánto esfuerzo toma aplicarla (esfuerzo en aprendizaje, instalación, configuración y ejecución)?	20
Total		**100**

PROYECTO 4

Descripción y caracterización del proyecto
GESTIÓN DE ALMACEN: Gestión de inventario en empresas logísticas: maquila para ofertas, ubicaciones de productos, entradas, salidas.

Caracterización		
Atributo	**Descripción**	**Valor**
Estático o dinámico	Tipo de técnica de testing requerida	Estática/Dinámica
Tipo de Software	Tipo de software a probar	Web, Mobile
Fase de ciclo de vida	Fase de desarrollo o ciclo de vida en el cual se realizarán las pruebas	Pruebas unitarias, Pruebas de integración de componentes, Pruebas del sistema, pruebas de integración del sistema, pruebas de aceptación
Escalabilidad	Tamaño del sistema a ser probado (LoC)	>3001
Entorno: lenguaje de programación	Lenguaje de programación usado para el desarrollo de software	C#
Entrada	¿Qué entradas están disponibles para el proceso de testing?	Código Fuente, Programa ejecutable, escenarios de ejecución, registros de ejecución, requisitos de la aplicación, valores superiores e inferiores de las variables.
Conocimiento	¿Qué conocimientos tiene el equipo de pruebas?	Dominio de aplicación, identificación de escenarios, Paradigma orientado a objetos, Codificación, generación de scripts de pruebas, trabajo en equipo, Liderazgo
Experiencia	¿Qué experiencia tiene el equipo de pruebas?	Alguna experiencia con pruebas de cobertura, Pruebas de requisitos, web testing
Salida	Salidas que el equipo espera producto de la aplicación de las técnicas de pruebas	Casos de prueba, Información de cobertura, Defectos, escenarios de pruebas
Tipos de defectos	Principales tipos de defectos que se quiere detectar	Interfaz, Algoritmo, Función, Timming/Serialización, Documentación
Aplicabilidad en tareas	Actividades de pruebas que se quiere realizar	Pruebas de cobertura de código, Pruebas de requisitos, Pruebas del sistema
Obtención de herramienta	Tipo de licenciamiento que prefiere para la herramienta de pruebas	Comercial

Preferencias del equipo de proyecto respecto al desempeño de las técnicas a recomendar

Atributo	**Descripción**	**Peso**
Completitud	Cobertura provista por los casos de prueba	20
Eficacia	Capacidad de encontrar defectos	20
Tamaño del banco de pruebas	Nivel de adecuación del número de casos de prueba generados por unidad de tamaño de software	08
Interacción	Cuán adecuados son los modos de interacción; (Por ejemplo: navegación, consultas, refinamiento sucesivo, etc.) soportados por la técnica	08
Guía o participación del usuario	Cuan apropiada es la guía o participación que la técnica requiere del usuario. Por ejemplo: Nada apropiada (la técnica es completamente manual), evaluación manual de las salidas, selección de entradas apropiadas, definición de patrones, filtrado, etc.	08
Fuentes de información	Disponibilidad de fuentes de información acerca de la técnica	05
Comprensibilidad	Si la técnica es o no es fácil de entender	15
Satisfacción subjetiva	Respecto a la técnica	06
Esfuerzo	¿Cuánto esfuerzo toma aplicarla (esfuerzo en aprendizaje, instalación, configuración y ejecución)?	10
Total		**100**

PROYECTO 5

Descripción y caracterización del proyecto

GESTION DE PRODUCCION - OPERARIOS: El sistema sirve para gestionar las tareas asignadas a los obreros y operarios, su estado y observaciones en el proceso de producción. Este sistema se integra con un sistema core que contiene los datos del personal, asi mismo este sistema produce información para el calculo de la producción y la gestion de los recursos humanos.

Caracterización		
Atributo	**Descripción**	**Valor**
Estático o dinámico	Tipo de técnica de testing requerida	Dinámica
Tipo de Software	Tipo de software a probar	Mobile
Fase de ciclo de vida	Fase de desarrollo o ciclo de vida en el cual se realizarán las pruebas	Pruebas de integración del sistema
Escalabilidad	Tamaño del sistema a ser probado (LoC)	501 – 1000
Entorno: lenguaje de programación	Lenguaje de programación usado para el desarrollo de software	Visual Basic
Entrada	¿Qué entradas están disponibles para el proceso de testing?	Código Fuente, Programa ejecutable, requisitos de la aplicación, valores superiores e inferiores de las variables.
Conocimiento	¿Qué conocimientos tiene el equipo de pruebas?	Identificación de escenarios, Paradigma orientado a objetos, Codificación ,Trabajo en equipo
Experiencia	¿Qué experiencia tiene el equipo de pruebas?	Pruebas de requisitos
Salida	Salidas que el equipo espera producto de la aplicación de las técnicas de pruebas	Casos de prueba, información de cobertura, escenarios de pruebas, Defectos
Tipos de defectos	Principales tipos de defectos que se quiere detectar	Interfaz, Algoritmo, Timming/Serialización, Documentación
Aplicabilidad en tareas	Actividades de pruebas que se quiere realizar	Pruebas de cobertura de código, Pruebas de requisitos, Pruebas del sistema
Obtención de herramienta	Tipo de licenciamiento que prefiere para la herramienta de pruebas	Comercial

Preferencias del equipo de proyecto respecto al desempeño de las técnicas a recomendar

Atributo	**Descripción**	**Peso**
Completitud	Cobertura provista por los casos de prueba	15
Eficacia	Capacidad de encontrar defectos	15
Tamaño del banco de pruebas	Nivel de adecuación del número de casos de prueba generados por unidad de tamaño de software	04
Interacción	Cuán adecuados son los modos de interacción; (Por ejemplo: navegación, consultas, refinamiento sucesivo, etc.) soportados por la técnica	10
Guía o participación del usuario	Cuan apropiada es la guía o participación que la técnica requiere del usuario. Por ejemplo: Nada apropiada (la técnica es completamente manual), evaluación manual de las salidas, selección de entradas apropiadas, definición de patrones, filtrado, etc.	07
Fuentes de información	Disponibilidad de fuentes de información acerca de la técnica	10
Comprensibilidad	Si la técnica es o no es fácil de entender	12
Satisfacción subjetiva	Respecto a la técnica	12
Esfuerzo	¿Cuánto esfuerzo toma aplicarla (esfuerzo en aprendizaje, instalación, configuración y ejecución)?	15
Total		**100**

PROYECTO 6

Descripción y caracterización del proyecto
INTEGRACION JUEGOS DE AZAR: Integración por servicios web entre plataforma de apuestas y proveedor de juegos de azar (gestión de billetera).

Caracterización		
Atributo	**Descripción**	**Valor**
Estático o dinámico	Tipo de técnica de testing requerida	Dinámica
Tipo de Software	Tipo de software a probar	Web
Fase de ciclo de vida	Fase de desarrollo o ciclo de vida en el cual se realizarán las pruebas	Pruebas unitarias Pruebas de integración de sistema Pruebas de aceptación
Escalabilidad	Tamaño del sistema a ser probado (LoC)	301 – 500
Entorno: lenguaje de programación	Lenguaje de programación usado para el desarrollo de software	PHP
Entrada	¿Qué entradas están disponibles para el proceso de testing?	Código Fuente Programa ejecutable Escenarios de ejecución Registros de ejecución
Conocimiento	¿Qué conocimientos tiene el equipo de pruebas?	Dominio de aplicación Paradigma orientado a objetos Trabajo en equipo Liderazgo
Experiencia	¿Qué experiencia tiene el equipo de pruebas?	Testing de requisitos
Salida	Salidas que el equipo espera producto de la aplicación de las técnicas de pruebas	Defectos
Tipos de defectos	Principales tipos de defectos que se quiere detectar	Función Timing/Serialization
Aplicabilidad en tareas	Actividades de pruebas que se quiere realizar	Pruebas unitarias, Pruebas del sistema
Obtención de herramienta	Tipo de licenciamiento que prefiere para la herramienta de pruebas	Open source

Preferencias del equipo de proyecto respecto al desempeño de las técnicas a recomendar

Atributo	**Descripción**	**Peso**
Completitud	Cobertura provista por los casos de prueba	0
Eficacia	Capacidad de encontrar defectos	30
Tamaño del banco de pruebas	Nivel de adecuación del número de casos de prueba generados por unidad de tamaño de software	20
Interacción	Cuán adecuados son los modos de interacción; (Por ejemplo: navegación, consultas, refinamiento sucesivo, etc.) soportados por la técnica	0
Guía o participación del usuario	Cuan apropiada es la guía o participación que la técnica requiere del usuario. Por ejemplo: Nada apropiada (la técnica es completamente manual), evaluación manual de las salidas, selección de entradas apropiadas, definición de patrones, filtrado, etc.	30
Fuentes de información	Disponibilidad de fuentes de información acerca de la técnica	0
Comprensibilidad	Si la técnica es o no es fácil de entender	0
Satisfacción subjetiva	Respecto a la técnica	0
Esfuerzo	¿Cuánto esfuerzo toma aplicarla (esfuerzo en aprendizaje, instalación, configuración y ejecución)?	20
Total		**100**

PROYECTO 7

Descripción y caracterización del proyecto

CONTROL DE TAREO: Sistema para gestionar condiciones de tareo de los obreros en distintos turnos en empresa minera. Registro diario faltas, vacaciones, descanso medico de los obreros, entre otros.

Caracterización		
Atributo	**Descripción**	**Valor**
Estático o dinámico	Tipo de técnica de testing requerida	Dinámica
Tipo de Software	Tipo de software a probar	Web
Fase de ciclo de vida	Fase de desarrollo o ciclo de vida en el cual se realizarán las pruebas	Pruebas unitarias Pruebas de sistema Pruebas de aceptación
Escalabilidad	Tamaño del sistema a ser probado (LoC)	301 – 500
Entorno: lenguaje de programación	Lenguaje de programación usado para el desarrollo de software	C#
Entrada	¿Qué entradas están disponibles para el proceso de testing?	Código Fuente, Programa ejecutable, Requisitos de aplicación Limites superiores e inferiores de las variables
Conocimiento	¿Qué conocimientos tiene el equipo de pruebas?	Definición y validación de funciones objetivo Conocimiento sobre como identificar el estado del sistema a probar Identificación de escenarios, Paradigma orientado a objetos Codificación, Trabajo en equipo, Liderazgo
Experiencia	¿Qué experiencia tiene el equipo de pruebas?	Testing de requisitos, Web testing
Salida	Salidas que el equipo espera producto de la aplicación de las técnicas de pruebas	Casos de prueba, Información de cobertura, Defectos
Tipos de defectos	Principales tipos de defectos que se quiere detectar	Chequeo, Algoritmo, Build/Package/Merge, Documentación Defectos
Aplicabilidad en tareas	Actividades de pruebas que se quiere realizar	Pruebas unitarias,Pruebas de requisitos, Pruebas del sistema
Obtención de herramienta	Tipo de licenciamiento que prefiere para la herramienta de pruebas	Open source

Preferencias del equipo de proyecto respecto al desempeño de las técnicas a recomendar

Atributo	**Descripción**	**Peso**
Completitud	Cobertura provista por los casos de prueba	0
Eficacia	Capacidad de encontrar defectos	20
Tamaño del banco de pruebas	Nivel de adecuación del número de casos de prueba generados por unidad de tamaño de software	0
Interacción	Cuán adecuados son los modos de interacción; (Por ejemplo: navegación, consultas, refinamiento sucesivo, etc.) soportados por la técnica	10
Guía o participación del usuario	Cuan apropiada es la guía o participación que la técnica requiere del usuario. Por ejemplo: Nada apropiada (la técnica es completamente manual), evaluación manual de las salidas, selección de entradas apropiadas, definición de patrones, filtrado, etc.	30
Fuentes de información	Disponibilidad de fuentes de información acerca de la técnica	10
Comprensibilidad	Si la técnica es o no es fácil de entender	0
Satisfacción subjetiva	Respecto a la técnica	0
Esfuerzo	¿Cuánto esfuerzo toma aplicarla (esfuerzo en aprendizaje, instalación, configuración y ejecución)?	30
Total		**100**

PROYECTO 8

Descripción y caracterización del proyecto

SOTESTER: Aplicación web que permite obtener recomendaciones de técnicas de testing de software para un proyecto específico. Permite registrar un proyecto, sus características, las preferencias acerca del desempeño de técnicas de testing y finalmente ofrece un ranking de técnicas recomendadas.

Caracterización		
Atributo	**Descripción**	**Valor**
Estático o dinámico	Tipo de técnica de testing requerida	Dinámica
Tipo de Software	Tipo de software a probar	Web
Fase de ciclo de vida	Fase de desarrollo o ciclo de vida en el cual se realizarán las pruebas	Pruebas de sistema
Escalabilidad	Tamaño del sistema a ser probado (LoC)	1001 – 3000
Entorno: lenguaje de programación	Lenguaje de programación usado para el desarrollo de software	C#
Entrada	¿Qué entradas están disponibles para el proceso de testing?	Código Fuente, Programa ejecutable
Conocimiento	¿Qué conocimientos tiene el equipo de pruebas?	Dominio de aplicación, Conocimiento acerca de cómo identificar el estado del sistema a probar, Paradigma orientado a objetos, Codificación ,Trabajo en equipo, Liderazgo
Experiencia	¿Qué experiencia tiene el equipo de pruebas?	--
Salida	Salidas que el equipo espera producto de la aplicación de las técnicas de pruebas	Casos de prueba, escenarios de pruebas, Defectos
Tipos de defectos	Principales tipos de defectos que se quiere detectar	Algoritmo, Función
Aplicabilidad en tareas	Actividades de pruebas que se quiere realizar	Pruebas unitarias, Pruebas del sistema
Obtención de herramienta	Tipo de licenciamiento que prefiere para la herramienta de pruebas	Open source

Preferencias del equipo de proyecto respecto al desempeño de las técnicas a recomendar

Atributo	**Descripción**	**Peso**
Completitud	Cobertura provista por los casos de prueba	10
Eficacia	Capacidad de encontrar defectos	30
Tamaño del banco de pruebas	Nivel de adecuación del número de casos de prueba generados por unidad de tamaño de software	10
Interacción	Cuán adecuados son los modos de interacción; (Por ejemplo: navegación, consultas, refinamiento sucesivo, etc.) soportados por la técnica	5
Guía o participación del usuario	Cuan apropiada es la guía o participación que la técnica requiere del usuario. Por ejemplo: Nada apropiada (la técnica es completamente manual), evaluación manual de las salidas, selección de entradas apropiadas, definición de patrones, filtrado, etc.	10
Fuentes de información	Disponibilidad de fuentes de información acerca de la técnica	10
Comprensibilidad	Si la técnica es o no es fácil de entender	5
Satisfacción subjetiva	Respecto a la técnica	5
Esfuerzo	¿Cuánto esfuerzo toma aplicarla (esfuerzo en aprendizaje, instalación, configuración y ejecución)?	15
Total		**100**

PROYECTO 9

Descripción y caracterización del proyecto
DETRACCIONES: Aplicativo que, en base a la facturación, permite el cálculo y generación de detracciones, permitiendo procesar su pago en las unidades de tesorería. regularización del pago de detracción.

Caracterización		
Atributo	**Descripción**	**Valor**
Estático o dinámico	Tipo de técnica de testing requerida	Dinámica
Tipo de Software	Tipo de software a probar	Web
Fase de ciclo de vida	Fase de desarrollo o ciclo de vida en el cual se realizarán las pruebas	Pruebas unitarias, Pruebas de sistema, Pruebas de regresión
Escalabilidad	Tamaño del sistema a ser probado (LoC)	301 – 500
Entorno: lenguaje de programación	Lenguaje de programación usado para el desarrollo de software	C#
Entrada	¿Qué entradas están disponibles para el proceso de testing?	Código Fuente, Programa ejecutable, Requisitos de aplicación
Conocimiento	¿Qué conocimientos tiene el equipo de pruebas?	Dominio de aplicación, Paradigma orientado a objetos, Trabajo en equipo, Liderazgo
Experiencia	¿Qué experiencia tiene el equipo de pruebas?	Testing de requisitos, Web testing
Salida	Salidas que el equipo espera producto de la aplicación de las técnicas de pruebas	Casos de prueba, Defectos
Tipos de defectos	Principales tipos de defectos que se quiere detectar	Chequeo, Algoritmo, Build/Package/Merge, Documentación
Aplicabilidad en tareas	Actividades de pruebas que se quiere realizar	Pruebas unitarias, Pruebas de requisitos, Pruebas del sistema
Obtención de herramienta	Tipo de licenciamiento que prefiere para la herramienta de pruebas	Open source

Preferencias del equipo de proyecto respecto al desempeño de las técnicas a recomendar

Atributo	**Descripción**	**Peso**
Completitud	Cobertura provista por los casos de prueba	10
Eficacia	Capacidad de encontrar defectos	20
Tamaño del banco de pruebas	Nivel de adecuación del número de casos de prueba generados por unidad de tamaño de software	0
Interacción	Cuán adecuados son los modos de interacción; (Por ejemplo: navegación, consultas, refinamiento sucesivo, etc.) soportados por la técnica	0
Guía o participación del usuario	Cuan apropiada es la guía o participación que la técnica requiere del usuario. Por ejemplo: Nada apropiada (la técnica es completamente manual), evaluación manual de las salidas, selección de entradas apropiadas, definición de patrones, filtrado, etc.	40
Fuentes de información	Disponibilidad de fuentes de información acerca de la técnica	10
Comprensibilidad	Si la técnica es o no es fácil de entender	0
Satisfacción subjetiva	Respecto a la técnica	0
Esfuerzo	¿Cuánto esfuerzo toma aplicarla (esfuerzo en aprendizaje, instalación, configuración y ejecución)?	20
Total		**100**

PROYECTO 10

Descripción y caracterización del proyecto
CONTROL DE MATERIALES EN PRODUCCIÓN: PLC en producción de etiquetas (troqueladoras). Calcular metros cuadrados de material que se consume en la máquina. Los datos se almacenan en un servidor y se consumen desde una aplicación web. La finalidad es determinar el tiempo de trabajo de la máquina y se contrasta con el trabajo reportado del operario.

Caracterización		
Atributo	**Descripción**	**Valor**
Estático o dinámico	Tipo de técnica de testing requerida	Dinámica
Tipo de Software	Tipo de software a probar	Batch
Fase de ciclo de vida	Fase de desarrollo o ciclo de vida en el cual se realizarán las pruebas	Pruebas unitarias
Escalabilidad	Tamaño del sistema a ser probado (LoC)	3001-500
Entorno: lenguaje de programación	Lenguaje de programación usado para el desarrollo de software	C#
Entrada	¿Qué entradas están disponibles para el proceso de testing?	Código Fuente, Programa ejecutable, escenarios de ejecución, requisitos de la aplicación, valores superiores e inferiores de las variables.
Conocimiento	¿Qué conocimientos tiene el equipo de pruebas?	Paradigma orientado a objetos, Codificación
Experiencia	¿Qué experiencia tiene el equipo de pruebas?	Pruebas de requisitos
Salida	Salidas que el equipo espera producto de la aplicación de las técnicas de pruebas	Información de cobertura, Defectos
Tipos de defectos	Principales tipos de defectos que se quiere detectar	Interfaz, Algoritmo, Timming/Serialización, Documentación
Aplicabilidad en tareas	Actividades de pruebas que se quiere realizar	Pruebas unitarias, Pruebas de cobertura de código, Pruebas del sistema
Obtención de herramienta	Tipo de licenciamiento que prefiere para la herramienta de pruebas	Comercial

Preferencias del equipo de proyecto respecto al desempeño de las técnicas a recomendar

Atributo	**Descripción**	**Peso**
Completitud	Cobertura provista por los casos de prueba	17
Eficacia	Capacidad de encontrar defectos	17
Tamaño del banco de pruebas	Nivel de adecuación del número de casos de prueba generados por unidad de tamaño de software	02
Interacción	Cuán adecuados son los modos de interacción; (Por ejemplo: navegación, consultas, refinamiento sucesivo, etc.) soportados por la técnica	10
Guía o participación del usuario	Cuan apropiada es la guía o participación que la técnica requiere del usuario. Por ejemplo: Nada apropiada (la técnica es completamente manual), evaluación manual de las salidas, selección de entradas apropiadas, definición de patrones, filtrado, etc.	10
Fuentes de información	Disponibilidad de fuentes de información acerca de la técnica	12
Comprensibilidad	Si la técnica es o no es fácil de entender	05
Satisfacción subjetiva	Respecto a la técnica	12
Esfuerzo	¿Cuánto esfuerzo toma aplicarla (esfuerzo en aprendizaje, instalación, configuración y ejecución)?	15
Total		**100**

Apéndice 4 – RECOMENDACIONES DE TÉCNICAS DE TESTING

PO	Recomendador	Recomendación	PO	Recomendador	Recomendación
0	E1	DT,ECP,FI	6	E1	FGN,NL,FT
0	E2	DCC,DT,DCH	6	E2	OA,FT,DTC
0	E3	DT,STD,FI	6	E3	CL,BC,FT
0	RS	BC,DCC,CW	6	RS	DCC,GM,OA
1	E1	DT,STD,APT	7	E1	DT,CW,SC
1	E2	DTC,DT,OA	7	E2	FC,DTC,DT
1	E3	ECP,DT,DCC	7	E3	BC,NL,CW
1	RS	CW,BC,FC	7	RS	BC,DCC,CW
2	E1	DTC,ECP,BVA	8	E1	DTC,DT,FI
2	E2	OA,DTC,DT	8	E2	DT,DCC,GM
2	E3	ECP,DTC,DT	8	E3	CL,BC,CW
2	RS	DCC,GM,BC	8	RS	BC,DCC,CW
3	E1	FGN,CW,FT	9	E1	ECP,CW,STD
3	E2	BC,ECP,SL	9	E2	OA,BC,DTC
3	E3	BC,APT,FT	9	E3	DCC,OA,DCH
3	RS	CW,OA,BC	9	RS	BC,DCC,FC
4	E1	BC,ECP,CW	10	E1	CL,FT,DCC
4	E2	DCC,FT,DT	10	E2	BC,FC,BVA
4	E3	ECP,DCC,FI	10	E3	DT,SC,FI
4	RS	CW,BC,DCC	10	RS	BC,OA,DCC
5	E1	CL,FI,STD			
5	E2	DTC,SC,BVA			
5	E3	DT,NL,FI			
5	RS	OA,DCC,CW			

Leyenda

.DT:Decision Tables	OA: Orthogonal Arrays
ECP:Equivalence Class Partitioning	FC: Function Coverage
FI:Formal Inspections	BVA: Boundary Value Analysis
DCC:Decision/Condition Coverage	GM: Graph Matrices
DCH:Desk checking	FGN: Flow Graph Notation
STD: State Transition Diagrams	FT:Fuzz Testing
BC:Branch coverage	SL:Simple Loops
CW: Code walkthrough	CL: Concatenated loops
APT: All Pairs Technique	SC: Statement coverage
DTC: Deriving Test Cases	NL: Nested loops